Microsystèmes de capteurs utilisant l'intelligence artificielle embarquée pour le domaine médical

par

Alexandre PERROTTON

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 31 MARS 2025

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Sylvain Cloutier, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Fabrice Vaussenat, codirecteur
Département de génie électrique à l'École de technologie supérieure

Mme. Rachel Bouserhal, présidente du jury
Département de génie électrique à l'École de technologie supérieure

M. Martin Bolduc, examinateur externe
Département de génie mécanique à l'Université du Québec à Trois-Rivières

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 27 MARS 2025

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier, avant toute chose, M. Sylvain Cloutier pour son encadrement, son écoute et son professionnalisme. Je souhaite également remercier M. Fabrice Vaussenat pour sa supervision et tous les bons repas partagés. Il m'a vraiment permis de prendre conscience de mes qualités et de ce dont j'étais capable. Je tiens à remercier tous mes collègues avec qui j'ai partagé des projets, de près ou de loin. Je remercie également ma famille, et plus particulièrement ma sœur, pour son soutien. Enfin, je tiens à remercier ma grande amie Julie pour sa bonne humeur et son soutien, sans qui je n'aurais pas réussi à finaliser ce mémoire.

Microsystèmes de capteurs utilisant l'intelligence artificielle embarquée pour le domaine médical

Alexandre PERROTTON

RÉSUMÉ

Le TinyML est un domaine émergent visant à développer des modèles d'apprentissage automatique compacts et économes en puissance de calcul et en énergie, adaptés aux dispositifs embarqués. Alors que les modèles de grande envergure requièrent d'importantes ressources de calcul et engendrent une empreinte environnementale non négligeable, le TinyML propose des solutions locales, rapides et sécurisées, ouvrant la voie à des applications innovantes dans des domaines tels que la santé et le suivi sportif.

Dans ce mémoire, je présente le développement d'un système innovant intégrant une matrice de capteurs imprimés appliquée sur un masque KN95, destiné à suivre en temps réel l'évolution de l'humidité relative à sa surface et, par extension, à analyser les schémas respiratoires des porteurs. Une encre sensible à l'humidité à base de ferrite de bismuth ($BiFeO_3$) a été imprimée par sérigraphie sur un circuit en argent, lequel a ensuite été cousu sur le masque et connecté à un système d'acquisition de données (Keithley DAQ6510).

Une série d'expérimentations, comprenant plusieurs cycles de respiration normale et profonde, a permis de constituer un jeu de données. Ces données ont été prétraitées et classifiées de manière non supervisée à l'aide de l'algorithme de clustering HDBSCAN, permettant d'identifier trois classes distinctes : respiration normale, respiration profonde et respiration irrégulière. La détection de passages en respiration irrégulière chez certains participants a ensuite été validée par un réseau de neurones, lequel a atteint une précision de 99.4% sur les données de test. Enfin, le modèle a été converti de TensorFlow vers TVM pour optimiser ses performances et assurer sa compatibilité avec une Arduino Nano 33 BLE, aboutissant à un temps d'inférence inférieur à 1ms.

Les résultats obtenus démontrent la faisabilité d'une solution TinyML fiable et autonome pour le suivi des schémas respiratoires en environnement embarqué.

Mots-clés: Apprentissage machine, TinyML, Masque KN95, Électronique imprimable, Arduino nano 33 BLE, capteur d'humidité, capteur imprimé

Microsystems of sensors using embedded artificial intelligence for the medical application

Alexandre PERROTTON

ABSTRACT

TinyML is an emerging field dedicated to the development of compact, computationally and energy-efficient machine learning models suitable for embedded devices. While large-scale models require extensive computational resources and incur a significant environmental footprint, TinyML offers local, fast, and secure solutions, paving the way for innovative applications in areas such as healthcare and sports monitoring.

In this thesis, I present the development of an innovative system that integrates a printed sensor matrix directly onto a KN95 mask, designed to monitor in real time the evolution of humidity on its surface and, by extension, to analyze the breathing patterns of its wearers. A humidity-sensitive ink based on bismuth ferrite ($BiFeO_3$) was screen-printed onto a silver circuit, which was then sewn onto the mask and connected to a data acquisition system (Keithley DAQ6510).

A series of experiments, comprising several cycles of normal and deep breathing, allowed us to build a dataset. These data were preprocessed and classified using an unsupervised approach with the HDBSCAN clustering algorithm, which enabled the identification of three distinct classes : normal breathing, deep breathing, and irregular breathing. The detection of irregular breathing episodes in certain participants was subsequently validated using a neural network, achieving an accuracy of 99.4% on the test data. Finally, the model was converted from TensorFlow to TVM to optimize its performance and ensure compatibility with an Arduino Nano 33 BLE, resulting in an inference time of less than 1 ms.

The obtained results demonstrate the feasibility of a reliable and autonomous TinyML solution for monitoring breathing patterns in an embedded environment.

Keywords: Machine learning, TinyML, Mask KN95, printed electronic , Arduino nano 33 BLE, humidity sensor, printed sensor

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ML	Machine Learning
SVM	Support Vector Machine
RIF	filtre à Réponse Impulsionnelle Finie
RII	filtre à Réponse Impulsionnelle Infinie
TFLite	TensorFlow Lite
GPU	Graphics Processing Unit
CPU	Central Processing Unit
TPU	Tensor Processing Unit
IA	Intelligence Artificielle
IoT	Internet of Things

INTRODUCTION

L'intelligence artificielle est devenue, en quelques années, l'un des domaines les plus dynamiques et médiatisés au monde. L'essor d'outils impressionnants accessibles à tous, tels que les robots conversationnels et les générateurs d'images, suscite à la fois fascination et inquiétudeNg *et al.* (2023). Ces outils, bien que puissants, nécessitent des ressources de plus en plus importantes pour évoluer. En effet, depuis quelques années, leurs besoins en ressources ont explosé, entraînant une augmentation exponentielle des coûts d'entraînement. Comme le présentent Maslej *et al.* (2024), entre 2018 et 2023, ces coûts sont passés d'une dizaine de milliers de dollars à plusieurs dizaines de millions. Cette escalade des coûts génère des critiques croissantes quant à l'impact environnemental de ces modèles, notamment en raison de l'empreinte carbone associée à leur entraînement et développement (Strubell, Ganesh & McCallum (2019)).

Ces modèles impressionnants ont souvent tendance à éclipser un autre type d'apprentissage automatique : l'apprentissage embarquéTsoukas, Boumpa, Giannakas & Kakarountas (2021). Contrairement à la plupart des systèmes de machine learning actuels qui nécessitent des capacités de calcul importantes et une connexion constante à des centres de données, l'apprentissage embarqué permet de traiter les données localement sur des dispositifs à faible consommation d'énergie (Schizas, Karras, Karras & Sioutas (2022)). Selon Warden & Situnayake (2019), ces nouveaux modèles de taille réduite offrent plusieurs avantages : indépendance vis-à-vis du réseau, meilleure sécurité des données, réduction de la latence et optimisation énergétique. Étant donné que toutes les données sont stockées localement, il n'est plus nécessaire d'envoyer des informations potentiellement sensibles vers des serveurs externes (Tsoukas *et al.* (2021)).

Bien que ces modèles embarqués soient moins puissants que leurs homologues déployés dans des centres de calcul, ils sont souvent suffisants pour les tâches qui leur sont assignées, optimisant ainsi la consommation énergétique pour ces types d'applications (Alajlan & Ibrahim (2022)). Cela permet également le développement des objets portables. Ces objets sont de plus en plus

utilisés pour des applications de santé et de suivi sportif. Banbury *et al.* (2021) souligne que cette approche constitue une solution idéale pour préserver la confidentialité tout en maintenant des performances élevées.

C'est autour de cette problématique que s'inscrit le TinyML, ou "petit apprentissage automatique". Ce nouveau champ d'études se concentre sur le développement de modèles compacts et efficaces, capables d'être embarqués sur des microcontrôleurs de manière autonome, économique, sécurisée et rapide (Han & Siebert (2022)).

L'objectif principal de cette recherche est de mettre en évidence les différentes techniques et outils développés pour le TinyML, avec un focus particulier sur l'exploration des synergies possibles entre les approches de réduction de la taille des modèles et celles dédiées à l'optimisation et à la réduction des transferts de données. En examinant ces combinaisons, cette étude vise à identifier les approches les plus prometteuses pour relever les défis du TinyML et favoriser son adoption à grande échelle.

Ce mémoire s'articule autour d'une expérimentation centrée sur la caractérisation et l'optimisation des masques KN95, tout en intégrant un suivi des schémas respiratoires des porteurs (O'Kelly, Arora, Pirog, Ward & Clarkson (2021)). Cette étude met en lumière les défis spécifiques liés à l'intégration de modèles TinyML dans des dispositifs embarqués pour des applications en santé. En particulier, elle se focalise sur trois aspects fondamentaux : la précision, l'autonomie énergétique et le traitement des données sur des dispositifs décentralisés, garantissant autonomie et confidentialité sans recours à un serveur distant.

Ce travail met en avant des techniques standardisées pour réduire la taille des modèles, tout en maintenant des performances analytiques adaptées à des tâches spécifiques. En élargissant le cadre de réflexion, ces approches complémentaires permettent de présenter des solutions variées pour relever les défis du TinyML.

Par ailleurs, l'intégration d'une matrice de capteurs embarqués associée à une intelligence artificielle dédiée à l'analyse des schémas respiratoires offre des avantages significatifs. En combinant des capteurs de haute précision et des algorithmes de traitement de données en temps réel, il devient possible de détecter et d'interpréter avec finesse les variations dans les patterns respiratoires, ce qui permet non seulement une surveillance continue et proactive de l'état de santé, mais également une réaction rapide en cas d'anomalies.

Ce dispositif, compact et autonome, présenterait un fort potentiel d'utilisation dans divers environnements. Il pourrait, par exemple, être déployé dans les établissements de santé et les hôpitaux afin d'assurer un suivi en temps réel des patients, notamment ceux présentant des pathologies respiratoires ou en post-opératoire. Par ailleurs, son déploiement dans des environnements industriels pourrait offrir la possibilité de surveiller la santé respiratoire des travailleurs, en particulier dans des milieux exposés à des polluants. En outre, ce dispositif pourrait également trouver sa place dans les domiciles et les centres de soins à domicile, en assurant une surveillance continue et discrète susceptible d'améliorer la qualité des soins tout en réduisant la nécessité de visites hospitalières (Koh, Ang, Ser & Tan (2021).

Enfin, cette recherche ambitionne de fournir des pistes solides pour guider les travaux futurs dans ce domaine émergent, en contribuant à la fois à son développement théorique et à son adoption pratique. L'approche intégrée présentée ici, alliant capteurs avancés et intelligence artificielle, ouvre la voie à des innovations majeures dans la surveillance de la santé, offrant des solutions pratiques et adaptables aux besoins spécifiques des utilisateurs.

CHAPITRE 1

APPRENTISSAGE MACHINE

L'intelligence artificielle (IA) est un domaine visant à créer des systèmes capables de simuler, modéliser ou d'imiter certaines formes d'intelligence humaine. Cela se traduit notamment par la capacité à reconnaître des motifs, à prendre des décisions et à apprendre. Cette dernière capacité, appelée apprentissage automatique (ou machine learning (ML)), est une sous-discipline de l'IA qui permet aux algorithmes de s'améliorer par eux-mêmes, sans intervention humaine directe.

Contrairement aux systèmes classiques, basés sur des règles explicitement programmées, le machine learning repose sur la capacité des modèles algorithmiques à détecter des schémas dans les données et à les interpréter pour faire des prédictions ou prendre des décisions.

Depuis son apparition dans les années 1950 (Rosenblatt (1958)), le machine learning est devenu un élément essentiel dans de nombreux domaines, tels que la santé, où il est utilisé pour des diagnostics médicaux, ou encore les transports, avec l'essor des véhicules autonomes. Il est également présent dans l'ingénierie sociale, à travers les algorithmes de recommandation ciblé, tels que ceux utilisés par Netflix ou Amazon (Song (2019)), et dans le domaine artistique, grâce aux modèles génératifs capables de créer des images, de la musique ou du texte. Cette évolution exponentielle repose principalement sur trois facteurs : l'augmentation massive des données disponibles, souvent désignée par le terme "Big Data" ; les avancées dans les architectures algorithmiques, comme les réseaux neuronaux profonds (Deep Learning) ; et l'amélioration des capacités matérielles, notamment avec l'utilisation de processeurs spécialisés tels que les processeurs graphiques (GPU) ou les unité de traitement de tenseurs (TPU) (Yue *et al.* (2024)).

L'apprentissage automatique peut résoudre une grande variété de problèmes grâce à trois approches principales : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Dans l'apprentissage supervisé, l'algorithme apprend à partir de données étiquetées, où chaque entrée est associée à une sortie. Il ajuste ses paramètres internes pour minimiser l'écart entre ses prédictions et les résultats attendus. Un exemple classique est la classification d'images pour distinguer des chiens et des chats. En revanche, l'apprentissage

non supervisé n'utilise pas de sorties étiquetées. Il regroupe les données en fonction de similarités ou en identifiant des structures cachées, ce qui le rend particulièrement utile pour détecter des anomalies, comme dans les transactions financières pour repérer des fraudes. Enfin, l'apprentissage par renforcement repose sur un système de récompenses. L'algorithme apprend par essais et erreurs en recevant des retours positifs ou négatifs en fonction de ses actions. C'est le cas de l'algorithme AlphaGo, qui s'est entraîné en jouant des millions de parties contre lui-même pour exceller au jeu de Go (Silver *et al.* (2016)).

Malgré ses nombreuses avancées, le machine learning fait face à plusieurs défis majeurs. Parmi eux, la qualité des données d'entraînement est cruciale, les performances des modèles dépendent directement de la quantité et de la qualité des données disponibles. Cependant, cela soulève des questions éthiques quant à la provenance et à l'utilisation des données, en particulier lorsque celles-ci contiennent des informations sensibles ou sont collectées sans consentement.

En outre, la complexité croissante des algorithmes exige des ressources informatiques toujours plus importantes, ce qui soulève des préoccupations environnementales. En effet, les infrastructures nécessaires pour entraîner et exécuter ces modèles consomment des quantités significatives d'énergie, posant ainsi la question de leur durabilité à long terme.

1.1 Mise en contexte

L'intelligence artificielle (IA), ainsi que ses sous-catégories que sont l'apprentissage machine et le deep learning, ont connu une évolution exponentielle au cours des dernières décennies. Depuis les premières théories formulées dans les années 1950 (Rosenblatt (1958)), les travaux autour de l'IA ont toujours eu pour objectif principal de doter les machines de capacités de réflexion proches de celles des humains. Ces avancées ont permis de réaliser des progrès significatifs dans des domaines tels que la reconnaissance de motifs, la prise de décision ou encore la résolution de problèmes complexes.

Le développement des méthodes modernes débute réellement dans les années 1980, avec l'introduction de l'algorithme de rétropropagation (Rumelhart, Hinton & Williams (1986)), qui

a marqué un tournant en offrant aux chercheurs un outil puissant pour optimiser les modèles d'apprentissage supervisé. Toutefois, ce n'est qu'à partir des années 2000, avec l'explosion du numérique et l'augmentation exponentielle des capacités de calcul, que l'apprentissage machine a véritablement pris son envol.

Les avancées matérielles, notamment la spécialisation des cartes graphiques (GPU) dans le traitement parallèle et matriciel, ont considérablement accéléré ces progrès (Nickolls, Buck, Garland & Skadron (2008)). Parallèlement, l'accès facilité à de vastes bases de données annotées a permis de développer des modèles d'apprentissage de plus en plus performants et complexes. Aujourd'hui, l'apprentissage machine est utilisé dans des domaines variés comme la santé, les transports, la finance, ou encore dans les appareils intelligents grand public. La recherche s'appuie de plus en plus sur ces algorithmes pour résoudre des problématiques complexes, autrefois inenvisageables. Parmi les exemples notables, on peut citer la détection précoce de maladies, le développement de nouveaux médicaments ou encore la conduite autonome. Cette accélération des progrès est en partie attribuable à l'implication croissante des entreprises privées en collaboration avec la recherche académique pour développer de nouveaux outils et algorithmes.

Enfin, l'accès à ces technologies s'est largement démocratisé grâce à l'émergence de frameworks et d'outils open source tels que Scikit-learn, PyTorch ou TensorFlow (Pedregosa, Varoquaux, Gramfort *et al.* (2011); Paszke, Gross, Massa *et al.* (2019); Abadi *et al.* (2016). Ces outils permettent aux développeurs de concevoir et d'implémenter des algorithmes intégrant les dernières avancées technologiques de manière simple et rapide.

L'effervescence générée par l'ensemble de ces facteurs a rendu l'apprentissage machine incontournable, tant pour résoudre des problématiques complexes que pour améliorer et optimiser les systèmes actuels.

1.2 Augmentation des données et complexité du traitement

Avec l'explosion du numérique au début des années 2000, le volume exponentiel de données générées, ainsi que leur accessibilité croissante, ont constitué un terrain propice à l'évolution des technologies d'apprentissage automatique. Chaque jour, une énorme quantité de données est produite via internet, notamment par les réseaux sociaux, les journaux en ligne, les pages web et bien d'autres sources. De plus, les capteurs IoT (Internet des objets) collectent en continu une grande variété de données, qu'il s'agisse de données météorologiques, énergétiques, biométriques ou encore d'images captées par des caméras (Albahri *et al.* (2021)). Les dispositifs médicaux représentent également une source importante de données. Les données biométriques, qu'elles proviennent d'images médicales ou de signaux physiologiques, sont riches et complexes. L'industrie, quant à elle, génère des volumes croissants de données, notamment pour le suivi des performances des machines, la maintenance prédictive ou l'optimisation des chaînes de production (Bhuiyan, Rahman, Billah & Saha (2021)).

Cette production exponentielle de données est souvent qualifiée de Big Data. Cependant, elle pose plusieurs défis majeurs. Tout d'abord, le volume de données collectées dépasse souvent les capacités de traitement des systèmes de calcul traditionnels. Ensuite, la grande variété de ces données (structurées, semi-structurées et non structurées) nécessite le développement d'algorithmes spécifiques capables de gérer cette diversité. Enfin, la vitesse à laquelle ces données sont générées dépasse parfois les capacités de traitement en temps réel, exigeant des solutions capables de traiter les flux de données instantanémentChen, Mao & Liu (2014).

Cependant, toutes les données collectées ne sont pas forcément exploitables. Des étapes avancées de prétraitement sont nécessaires pour nettoyer les données du bruit, des valeurs manquantes ou des biais introduits lors de leur collecte. Ces traitements rendent les phases de prétraitement de plus en plus complexes et coûteuses à mesure que la quantité et la diversité des données augmentent (Sun & Han (2012)). La croissance exponentielle des données a également mis en lumière les limites des modèles traditionnels comme les régressions linéaires ou les arbres de décision. C'est dans ce contexte que les algorithmes d'apprentissage profond (deep learning) ont

émergé. Ces derniers permettent d'exploiter ces vastes ensembles de données et de capturer des caractéristiques abstraites et complexes qui seraient inaccessibles aux approches traditionnelles (Alonso, Sieber & Zeilinger (2024)).

Parallèlement, la taille des modèles modernes, comme GPT ou DALL-E 2, a considérablement augmenté. Ces modèles reposent sur des architectures de réseaux neuronaux profonds appelées transformeurs (transformers), qui sont particulièrement efficaces pour traiter des séquences de données et modéliser des relations complexes entre les mots. GPT fait partie des grands modèles de langage (LLM, pour Large Language Models), capables de comprendre et de générer du texte de manière fluide en s'appuyant sur de vastes corpus de données. Ces modèles, qui contiennent plusieurs milliards de paramètres, nécessitent des ressources matérielles adaptées pour être entraînés. Un simple ordinateur personnel est insuffisant : leur entraînement requiert des serveurs de calcul spécialisés et peut durer plusieurs mois. Les coûts associés à ces processus d'entraînement sont élevés, atteignant parfois plusieurs centaines de millions de dollars, et la consommation énergétique est considérable, soulevant des questions sur leur impact environnemental (Frosio (2023)).

Un autre défi majeur est que ces modèles nécessitent souvent des données annotées pour orienter l'apprentissage et obtenir des résultats précis. Or, l'annotation manuelle de millions de données est une tâche longue et coûteuse. C'est pourquoi des approches d'apprentissage non supervisé ou auto-supervisé sont de plus en plus utilisées. Ces méthodes tentent d'extraire automatiquement des caractéristiques significatives des données, sans intervention humaine, ce qui rend l'opération plus rapide, mais également plus complexe à optimiser. Ainsi, une tendance émerge à combiner apprentissage supervisé, pour guider le modèle, et apprentissage non supervisé, pour le cœur de l'entraînement (Chen *et al.* (2014)).

Enfin, on observe l'émergence de modèles d'apprentissage automatique spécialisés dans l'entraînement d'autres modèles de deep learning plus complexes.

Ces différentes avancées ouvrent la voie à des applications en temps réel et à des systèmes pouvant être déployés sur des appareils à faible consommation énergétique, tout en respectant des

réglementations de plus en plus strictes concernant l'utilisation et l'éthique de ces technologies. Toutefois, cela reste un défi majeur que la recherche s'efforce activement de résoudre, en explorant des approches innovantes pour concilier performance, durabilité et conformité aux normes en vigueur.

1.2.1 Domaine médical

Le domaine médical est l'un des secteurs les plus en développement grâce aux avancées du machine learning. En effet, l'un de ses principaux avantages est le diagnostic précoce des maladies. Les données médicales font partie des informations les plus complexes à interpréter, et il est possible que certaines pathologies passent inaperçues, notamment lorsque la maladie est à un stade très précoce (Wu, Wang, Su & Hsieh (2022); Nahavandi, Alizadehsani & Khosravi (2022)).

L'intelligence artificielle (IA) permet aujourd'hui d'assister les médecins dans la détection de certaines maladies, même lorsque les informations disponibles sont limitées. Par exemple, il a été démontré que des algorithmes d'apprentissage automatique améliorent significativement la détection du cancer du sein, avec une précision équivalente, voire supérieure, à celle des radiologues (Sabry, Eltaras & Labda (2022); Shah & Khan (2021)). Ces systèmes permettent de détecter des risques plusieurs années avant que des symptômes cliniques ne se manifestent (Yin & Jha (2017)).

De plus, les algorithmes jouent désormais le rôle de véritables assistants médicaux, permettant aux professionnels de santé de poser des diagnostics plus précis et rapides en se basant sur une liste de symptômes (Yin & Jha (2017); Saad, Zaki & Abdelsalam (2024)). Cette évolution contribue à la réduction des erreurs médicales et à l'amélioration des soins aux patients.

Une avancée majeure réside dans l'auto-diagnostic rendu possible par les dispositifs portables. Grâce aux algorithmes embarqués, il est aujourd'hui possible pour le grand public de suivre ses données biologiques en temps réel (Nahavandi *et al.* (2022); Awotunde, Folorunso & Bhoi (2021)). Bien que ces méthodes ne puissent remplacer un diagnostic médical, elles facilitent

l'accès à une surveillance minimale de la santé. Les capteurs portables, confrontés à des mouvements parasites, exploitent les techniques de machine learning pour optimiser les mesures et garantir des données fiables et exploitables (He & Zhao (2021)).

L'intégration croissante de l'intelligence artificielle dans la médecine démontre son potentiel à transformer le diagnostic, la prévention et la prise en charge des patients.

1.2.2 Domaine agricole

Le domaine agricole est en constante évolution, que ce soit pour l'amélioration de la croissance des plantations ou pour l'entretien des sols. Depuis quelques années, de nouvelles solutions utilisant des algorithmes d'apprentissage machine ont été développées pour assurer la surveillance et l'entretien ciblé des plantations (Abbas, Zhang, Zheng, Alami & Alrefaei (2023); Maraveas (2022)).

L'une des technologies qui évoluent le plus rapidement est celle des drones de surveillance. Équipés de caméras et de capteurs, ces drones peuvent analyser l'état de santé des cultures, évaluer leurs besoins en eau et détecter la présence de parasites (Akbar, Kamarulzaman & Muzahid (2024)). Pour y parvenir, les drones sont souvent munis de caméras haute résolution, ainsi que de caméras thermiques ou hyperspectrales permettant une analyse plus précise des maladies et du stress hydrique (Elvanidi & Katsoulas (2022)).

Cette technologie permet d'obtenir des données précises sur les besoins en eau des champs, ce qui optimise la consommation d'eau, en particulier dans les régions les plus arides. Par exemple, les algorithmes d'apprentissage profond sont capables de détecter les anomalies d'irrigation et les maladies en temps réel (Khan, Ibrahim & Zeki (2020a)). De plus, elle facilite la détection précoce de la propagation d'un parasite ou d'un virus. Les drones pulvérisateurs peuvent alors intervenir localement pour traiter la zone affectée, ce qui réduit l'impact des parasites sur la récolte (Ali, Shaikh & Hasan (2024)). Toutefois, ces procédés restent encore onéreux et sont pour l'instant peu répandus dans les exploitations agricoles de petite taille (Aslan, Durdu & Sabanci (2022)).

D'autres méthodes sont actuellement en cours de développement pour réduire les coûts et rendre ces technologies plus accessibles. Une des solutions proposées concerne l'utilisation des serres agricoles. Les serres, nécessitant un entretien particulier et offrant un accès simplifié aux plantes, permettent plus facilement d'implémenter des systèmes automatiques capables de détecter le développement de maladies à l'intérieur (Dubey, Sindhwani & Anand (2021)). Contrairement aux drones, qui interviennent sur des zones entières, ces méthodes ciblent les plantes individuellement, tout en utilisant des dispositifs connectés et des capteurs IoT pour surveiller l'environnement des cultures (Singh & Rokade (2021)).

Par ailleurs, des modèles de machine learning avancés appliqués aux serres permettent non seulement de prédire les besoins en nutriments et en eau des plantes, mais aussi de réguler automatiquement les conditions environnementales pour optimiser la croissance des cultures (Shaikh, Rasool & Lone (2022)). Cette automatisation contribue à la productivité tout en minimisant les ressources utilisées, ce qui est essentiel pour un avenir agricole durable.

1.3 Solution apportée par les algorithmes d'apprentissage machine

1.3.1 Détection d'anomalies

La détection d'anomalies est une méthode visant à identifier des comportements, des schémas ou des observations qui dévient d'un comportement considéré comme normal.

La méthode la plus simple repose sur des règles basées sur des seuils : toute valeur dépassant un certain seuil est considérée comme une anomalie. Bien que cette approche soit facile à implémenter, elle peut manquer de flexibilité face à des données plus complexes (Sinha, Stavrakis, Simić & Stojanović (2023)).

Une méthode plus avancée consiste à utiliser des modèles d'apprentissage supervisé. Ces modèles sont très efficaces pour détecter les anomalies déjà présentes dans le jeu de données d'entraînement. Cependant, ils restent limités lorsqu'il s'agit de repérer des anomalies inédites, non fournies durant l'apprentissage (Cen *et al.* (2017)).

Enfin, les méthodes non supervisées permettent de détecter des anomalies sans nécessiter d'étiquetage préalable des données. Elles sont particulièrement efficaces pour découvrir des anomalies inconnues, mais elles ne permettent pas de déterminer la nature précise de l'anomalie. Ces algorithmes peuvent donc nécessiter une seconde étape d'analyse pour identifier le type d'anomalie détectée (Truong *et al.* (2017)).

Parmi les méthodes d'apprentissage non supervisé, le clustering est l'une des plus simples et efficaces pour mettre rapidement en évidence des anomalies. Les données sont d'abord pré-traitées pour extraire des caractéristiques pertinentes, telles que la moyenne, la variance, l'écart-type, la température ou la distance. Ensuite, elles sont regroupées en clusters à l'aide d'algorithmes comme K-Means.

L'algorithme K-Means divise les données en un nombre prédéfini de groupes (k clusters) en minimisant la distance entre les points et les centres de chaque cluster. Chaque point est assigné au cluster dont le centre est le plus proche, et les centres sont ajustés itérativement jusqu'à stabilisation. Toute donnée se trouvant trop éloignée du centre d'un cluster est alors considérée comme une valeur anormale (Stewart & Al-Khassaweneh (2022)). Cependant, K-Means suppose que les clusters sont sphériques et de taille similaire, ce qui peut limiter sa performance sur des données de structure complexe.

Pour surmonter certaines de ces limites, HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) est une alternative robuste. Cet algorithme repose sur la densité des points : il regroupe les données en identifiant des régions de forte densité séparées par des zones de faible densité. Contrairement à K-Means, HDBSCAN n'exige pas de spécifier le nombre de clusters à l'avance et peut détecter des clusters de formes variées.

1.3.2 Analyse d'image

1.3.2.1 Modèle de segmentation

Les modèles de segmentation sont basés sur des réseaux de neurones à convolution (CNN). Leur objectif est de segmenter des parties spécifiques d'une image, que ce soit un objet distinct, une zone de couleur particulière ou une structure complexe. Ces modèles sont couramment utilisés pour la détection d'objets et la reconnaissance d'entités visuelles dans des domaines varié (Xu, Wang, Chen & Li (2015)).

Un des pipelines les plus efficaces pour la segmentation et la reconnaissance d'objets combine plusieurs modèles : ResNet, Faster R-CNN et Mask R-CNN (Huang *et al.* (2020)).

- **ResNet (Réseau Résiduel) :** c'est un réseau de neurones convolutionnel conçu pour extraire des caractéristiques riches et profondes à partir des images. Il agit comme un prétraitement en produisant une carte des caractéristiques (feature map) de l'image source. Cette carte contient les informations essentielles nécessaires pour la segmentation et la détection des objetsTarg, Almeida & Lyman (2016).
- **Faster R-CNN :** Faster R-CNN prend ensuite cette carte des caractéristiques et la transmet à un Réseau de Proposition de Régions (Region Proposal Network ou RPN). Ce réseau génère des régions d'intérêt (Regions of Interest ou RoIs) susceptibles de contenir des objets. Ces zones sont ensuite évaluées pour déterminer la probabilité qu'elles contiennent effectivement un objet, et les coordonnées des boîtes englobantes sont ajustées pour s'adapter précisément aux objets détectés. Les RoIs sont enfin classifiées selon des catégories prédéfinies, comme celles du jeu de données COCOBharati & Pramanik (2020).
- **Mask R-CNN :** Mask R-CNN est une extension de Faster R-CNN, qui ajoute une branche supplémentaire pour générer des masques de segmentation pour chaque objet détecté. Ces masques permettent de délimiter précisément les contours des objets au pixel près, offrant ainsi des résultats de segmentation plus détaillésBharati & Pramanik (2020).

Les modèles comme Faster R-CNN sont souvent employé pour la détection d'objets dans des contextes tels que les véhicules autonomes ou la vidéosurveillance. Mask R-CNN, quant à lui, est particulièrement utile pour des tâches nécessitant une segmentation précise, comme la segmentation d'images médicales ou l'analyse de la santé des plantes en agriculture.

1.3.2.2 Segment Anything Model (SAM)

Le Segment Anything Model (SAM), développé par Meta, représente une avancée majeure dans le domaine de la segmentation universelle. Il s'agit d'un modèle capable de segmenter tous types d'objets sans nécessiter d'entraînement spécifique supplémentaire, ce qui le distingue des modèles traditionnels nécessitant des ajustements ou des données annotées adaptées à des tâches spécifiques (Yi, Liu, Chen, Zhang & Fan (2023)). Contrairement à des modèles comme Faster R-CNN, SAM utilise une architecture plus moderne basée sur le Vision Transformer (ViT). Cette conception lui permet de fournir directement des masques précis sans passer par des étapes intermédiaires, simplifiant ainsi le pipeline de segmentation.

1.3.2.3 Combinaison des modèles

Bien que SAM soit performant pour générer des masques universels, il ne remplace pas totalement des modèles comme Faster R-CNN ou Mask R-CNN. Cependant, ces modèles peuvent être combinés à SAM pour exploiter leurs forces respectives. En associant ces différentes approches, il devient possible de réaliser des tâches avancées telles que : la segmentation guidée par la détection d'objets, la classification de segments, ou encore l'affinage rapide de la segmentationBharati & Pramanik (2020). Cette combinaison permet d'améliorer la précision et la flexibilité des modèles en tirant parti des atouts spécifiques de chacun : la capacité de SAM à générer des masques précis à partir de simples indications et la robustesse des modèles comme Faster R-CNN et Mask R-CNN pour la détection et la classification d'objets (Shu, Nian, Yu & Li (2020)). On retrouve ce type de synergie dans plusieurs domaines :

- **Médical :** Détection et segmentation de tumeurs ou d'organes sur des images médicales complexes.
- **Agriculture :** Identification des zones affectées par des maladies à partir d'images aériennes de champs.
- **Vision industrielle :** Tri d'objets sur des chaînes de production, inspection de défauts sur des produits manufacturés.

1.4 Techniques de prétraitement des données

Les données obtenues directement via des capteurs, des images ou des bases de données peuvent ne pas être immédiatement adaptées à une analyse ou à une modélisation. Elles peuvent être bruitées, hétérogènes, complexes ou trop volumineuses, nécessitant ainsi des étapes de prétraitement pour les rendre exploitables. Le prétraitement des données permet de nettoyer et de normaliser les données pour éliminer les erreurs et les incohérences. Il facilite aussi l'extraction d'informations pertinentes, optimisant ainsi les performances des modèles de prédiction. Enfin, il contribue à la réduction de la taille en mémoire des données, ce qui est crucial pour les systèmes embarqués et les applications en temps réel. Un bon prétraitement améliore non seulement la qualité des données, mais aussi l'efficacité et la précision des modèles d'apprentissage automatiqueAlshdaifat, Alshdaifat, Alsarhan, Hussein & El-Salhi (2021).

1.4.1 Standardisation et normalisation des variables

L'une des méthodes de bases pour le prétraitement des données est la standardisation et normalisation des données. Ces méthodes permettent de rendre les données comparables et plus adaptées à l'interprétation par les modèles d'apprentissage automatique. En effet, des variables avec des échelles ou des magnitudes trop disparates peuvent nuire aux performances des algorithmes, notamment ceux reposant sur des métriques de distance. Les modèles peuvent accorder un poids disproportionné aux variables avec de grandes valeurs numériques,

cela peut biaiser les résultats et entraîner une perte de précision (Klambauer, Unterthiner, Mayr & Hochreiter (2017)).

La standardisation consiste à transformer les variables pour qu'elles aient une moyenne nulle et un écart-type unitaire. Cela est particulièrement utile lorsque les données suivent une distribution gaussienne ou pour les modèles sensibles aux variations des échelle, tels que les algorithmes de régression linéaire, les Support Vector Machine (SVM), et les réseaux de neurones (li & Liu (2011)). La formule pour standardiser les variables est :

$$x_{\text{standardisé}} = \frac{x - \mu}{\sigma} \tag{1.1}$$

où μ est la moyenne de la variable, σ est l'écart-type.

La normalisation quant à elle transforme les variables pour les limitées à une plage spécifique, généralement entre 0 et 1. Cela est particulièrement utilisé pour les données ayant une distribution arbitraire et pour les modèles nécessitant une entrée bornée comme les algorithmes de clustering ou les réseaux neuronaux utilisant des fonctions d'activation sensibles aux échelles (Virmani, Taneja & Malhotra (2015)). La formule pour normaliser les données est :

$$x_{\text{normalisé}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1.2}$$

où $x_{\min}$ et $x_{\max}$ représentent les valeurs minimale et maximale du jeu de donné.

1.4.2 Filtrage et réduction du bruit

La plupart du temps les données obtenues directement à partir de capteurs, comme des signaux ou des images, peuvent être bruitées ou imparfaites. Un signal bruité peut biaiser les prédictions en apportant des informations non pertinentes à nos modèles et ainsi réduire les performances

globales. Il est donc essentiel de passer par une phase de filtration et de réduction du bruit dans nos données.

1.4.2.1 Filtres passe-bas

Le filtre passe-bas est une méthode qui permet de supprimer les hautes fréquences d'un signal tout en gardant les basses fréquences. Il est principalement utilisé pour nettoyer les données des oscillations rapides pouvant être dues à des mouvements brusques ou des erreurs, et à réduire le bruit aléatoire.

Les filtre passes-bas numériques peuvent être réalisés de deux façons différentes en utilisant une réponse impulsionnelle finie (RIF) ou une réponse impulsionnelle infinie (RII)Alessio (2016).

Les filtres RIF passe-bas sont calculés à partir de l'équation suivante :

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k] \tag{1.3}$$

où $y[n]$ est le signal filtré à l'instant n, $x[n]$ est le signal d'entrée à l'instant n et b_k sont les coefficients de la fonction du transfert du filtre. N est l'ordre du filtre.

Les coefficients b_{k} du filtre sont calculés pour laisser passer les fréquences inférieures à la fréquence de coupure f_{c}.

Le filtre par moyenne mobile est l'un des filtres RIF les plus simples et permet de lisser les signaux discrets. Il consiste à remplacer chaque valeur par la moyenne des valeurs d'une fenêtre glissante autour de lui (Mou & Duhamel (1987)). Sa formule est la suivante :

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \tag{1.4}$$

où $y[n]$ est le signal filtré à l'instant n, $x[n]$ est le signal d'entrée à l'instant n, M est la taille de la fenêtre glissante.

Le filtre passe-bas par fenêtre glissante est équivalent à un filtre RIF où tous les coefficients sont égaux à $\frac{1}{M}$.

Les filtres RIF sont les filtres les plus utilisés car ils sont stables, simples à concevoir et idéaux pour le prétraitement de données. Cependant, ils demandent une complexité de calcul plus élevée, ce qui n'est pas un problème lors de l'entraînement initial d'un modèle, mais peut l'être lors de l'inférence sur des cibles embarquées. C'est pour cela que les filtres RII, moins stables, et plus complexes à concevoir, sont préférés pour les applications TinyML. Ils sont plus efficaces en termes de calcul, permettant d'utiliser moins de ressources sur les systèmes embarqués. Les filtres RII sont donc parfaitement adaptés au nettoyage des données directement en sortie des capteurs (Mou & Duhamel (1987)).

Les filtres RII sont des filtres récursifs, c'est-à-dire qu'ils utilisent les valeurs passées et présentes du signal d'entrée, ainsi que les valeurs passées de la sortie. Ils sont l'équivalent des filtres à boucle de réaction en fonctionnement continu (Alessio (2016)). Leur équation est donnée sous la forme :

$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i] - \sum_{j=1}^{M} a_j \cdot y[n-j] \tag{1.5}$$

où $y[n]$ est le signal filtré à l'instant n, $x[n]$ est le signal d'entrée à l'instant n, b_i les coefficients des termes de l'entrée, a_j les coefficients des termes de rétroaction, N l'ordre du filtre de l'entrée, M l'ordre du filtre de rétroaction.

Il est important de noter que la complexité des filtres RII est plus grande que les filtres RIF. Cependant, un filtre RII nécessite un ordre plus faible pour avoir des performances équivalentes au filtre RIF, le rendant rentable en termes de ressources utilisées.

1.4.2.2 Filtres récursifs adaptatifs

En plus des approches RIF et RII classiques, il existe des filtres récursifs adaptatifs, très utilisés dans le domaine médical pour le traitement de signaux physiologiques (électrocardiogramme (ECG), électroencéphalogramme (EEG), etc.). Contrairement aux filtres à coefficients fixes, les filtres adaptatifs mettent à jour leurs coefficients en temps réel, en fonction des caractéristiques du signal observé. Cela leur permet de s'adapter aux variations rapides et aux artefacts spécifiques à l'environnement ou au patient, améliorant ainsi la qualité du signal filtré (Haque, Kaiser & Aditya (2005)).

Les algorithmes LMS (Least Mean Squares) et RLS (Recursive Least Squares) sont particulièrement utilisé pour leur efficacité dans la réduction de bruits, comme ceux liés aux interférences électriques ou aux mouvements brusques. L'ajustement continu des coefficients optimise la convergence vers la meilleure réponse filtrante au fur et à mesure de l'acquisition du signal.

Les filtres récursifs adaptatifs trouvent ainsi leur pertinence dans des tâches telles que la suppression du bruit de fond sur un signal d'électrocardiogramme (ECG) ou l'élimination d'artefacts de mouvement en électroencéphalographie (EEG) (Mugdha, Rawnaque & Ahmed (2015)). Leur capacité à s'ajuster de manière dynamique aux caractéristiques changeantes des signaux médicaux en fait un outil incontournable pour la fiabilité de l'analyse et l'optimisation des performances des modèles de prédiction.

1.4.2.3 Transformée de Fourier Rapide (FFT)

La transformée de Fourier est une méthode d'analyse fréquentielles des signaux. Elle permet de décomposer un signal temporel en une multitude de vaguelettes composant le signal de base, faisant passer le signal du domaine temporel au domaine fréquentiel. La transformée de Fourier rapide est une version optimisée de la transformée de Fourier discrète (DFT), ce qui réduit la complexité des calculsSzedo, Yang & Dick (2001).

La DFT est donnée par :

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1 \tag{1.6}$$

où $X[k]$ correspond aux coefficients fréquentiels, $x[n]$ est le signal d'entrée dans le domaine temporel , N est le nombre total de points du signal, k est l'indice de la fréquence.

La FFT permet de retirer facilement le bruit d'un signal en le décomposant en ses composantes fréquentielles. Cette décomposition facilite l'extraction des fréquences les plus significatives, ce qui permet d'isoler les caractéristiques importantes du signal tout en éliminant le bruit (Plonka, Potts, Steidl & Tasche (2018)).

La FFT est un excellent outil de prétraitement des signaux temporels. En une seule opération, elle permet non seulement de filtrer le bruit, mais aussi d'extraire et de compresser les données, en conservant uniquement les caractéristiques les plus pertinentes (Plonka *et al.* (2018)).

1.5 Frameworks pour l'apprentissage machine

L'architecture des différents types d'algorithmes, ainsi que leur implémentation, peuvent s'avérer complexes si elles doivent être programmées depuis zéro. Toutefois, de nombreux outils existent pour faciliter le développement de modèles d'apprentissage machine, ainsi que l'application des différentes méthodes d'optimisation et d'entraînement. Ces outils, appelés *frameworks*, simplifient la programmation en offrant un accès à toutes les fonctions nécessaires, à des paramètres variés et des modèles préconçus (Abadi *et al.* (2016)). Ils permettent ainsi de développer des modèles performants en utilisant les dernières avancées technologiques issues de la recherche. De plus, ces frameworks incluent par défaut des outils d'optimisation, comme Adam, ainsi que des techniques de validation, telles que la validation croisée, pour améliorer la précision des modèles (Pedregosa *et al.* (2011)).

1.5.1 Scikit-learn

Scikit-learn est une bibliothèque incontournable pour l'apprentissage machine traditionnel. Bien qu'elle ne supporte pas l'apprentissage profond, elle permet d'implémenter facilement des algorithmes de base couramment utilisés dans diverses applications et propose des algorithmes classiques tels que la régression linéaire, les arbres de décision ou encore les machines à vecteurs de support (SVM).

Scikit-learn offre des fonctionnalités avancées telles que la validation croisée pour évaluer la robustesse des modèles, l'optimisation des hyperparamètres via des méthodes comme *GridSearchCV* et *RandomizedSearchCV*, et des outils de prétraitement des données, notamment pour la normalisation, l'encodage des variables catégorielles et la réduction de dimension. Sa simplicité d'utilisation en fait un choix idéal pour les phases initiales du développement ou pour des cas où l'apprentissage profond serait excessif (Pedregosa *et al.* (2011)).

1.5.2 TensorFlow

TensorFlow est un framework de référence dans le domaine de l'apprentissage automatique et de l'apprentissage profond, développé initialement par Google. Il se distingue par sa capacité à fonctionner sur une large gamme de plateformes, depuis les serveurs de calcul jusqu'aux dispositifs embarqués, grâce à ses nombreuses optimisations matérielles. Historiquement, TensorFlow reposait sur un graphe computationnel statique, ce qui facilitait l'optimisation et le déploiement à grande échelle (Abadi *et al.* (2016)).

Depuis la version 2.0, TensorFlow a toutefois introduit l'exécution "eager" par défaut (Eager Execution), rendant le développement et le débogage plus intuitifs, tout en préservant la possibilité de convertir le code en graphe pour des besoins de performance ou de production. De plus, l'écosystème s'est élargi avec TensorFlow Lite pour le déploiement sur plateformes mobiles et systèmes embarqués.

1.5.3 Keras

Keras est une interface de haut niveau conçue pour simplifier la création et le prototypage rapide de modèles de réseaux de neurones. À l'origine, Keras était compatible avec plusieurs moteurs de calcul (Theano, CNTK, etc.), mais depuis l'arrivée de TensorFlow 2.0, elle est devenue l'API de haut niveau officiellement intégrée à TensorFlow. Cela signifie que les utilisateurs peuvent profiter de la simplicité syntaxique de Keras tout en accédant simultanément aux fonctionnalités avancées de TensorFlow.

Le principal avantage de Keras réside dans sa facilité d'utilisation. Sa syntaxe claire permet de mettre au point des prototypes de manière rapide et itérative. Les modèles développés avec Keras peuvent être convertis directement vers TensorFlow Lite afin d'être déployés sur des dispositifs embarqués. Cette capacité est cruciale pour la mise en œuvre de réseaux de neurones sur des microcontrôleurs tels que l'Arduino Nano 33 BLE ou encore sur des appareils à faible puissance comme le Raspberry Pi Zero. Keras offre ainsi une solution pratique et performante pour la vision par ordinateur embarquée et les applications exigeant une consommation de ressources minimale (Chollet (2015)).

1.5.4 PyTorch

PyTorch est un framework d'apprentissage profond très apprécié dans le domaine de la recherche, grâce à son architecture dynamique et sa flexibilité. Contrairement à TensorFlow, PyTorch utilise des graphes computationnels dynamiques, ce qui facilite le débogage et l'expérimentation (Paszke *et al.* (2019)). PyTorch est particulièrement pertinent pour des solutions nécessitant une grande flexibilité dans l'optimisation des modèles, notamment dans des applications de robotique et d'IoT industriel.

CHAPITRE 2

APPRENTISSAGE MACHINE EMBARQUÉ

L'apprentissage automatique embarqué consiste à concevoir et optimiser des modèles pour qu'ils puissent tourner sur des dispositifs aux ressources limitées, souvent contraints en termes de mémoire et de puissance de calcul. Cela rend impossible le déploiement de modèles complexes et volumineux sur ces cibles. Cependant, plusieurs techniques permettent de réduire la taille de modèles complexes, permettant ainsi de réaliser des prédictions localement, directement sur l'appareil. L'intérêt principal de l'embarquement d'un modèle d'apprentissage automatique est de pouvoir effectuer des prédictions sur le dispositif lui-même, sans dépendre d'un système externe ou puissant pour réaliser les calculs (Ray (2022b)).

2.1 TinyML

TinyML, ou "petit apprentissage machine", désigne une branche de l'intelligence artificielle spécifiquement dédiée au déploiement de modèles d'apprentissage automatique sur des dispositifs à faible puissance de calcul et à ressources limitées (Warden & Situnayake, 2019). Ces dispositifs incluent les microcontrôleurs, les systèmes de capteurs et autres systèmes électroniques simples (Banbury *et al.*, 2021). TinyML rend possible l'intégration de capacités d'IA dans des appareils extrêmement compacts, tout en répondant à des contraintes strictes en termes de consommation énergétique et de coûts (Ghahramani *et al.*, 2021).

TinyML repose sur quatre principes fondamentaux : l'autonomie des systèmes, la rapidité d'inférence, la faible consommation d'énergie et la sécurité des données (Fedorov, Burda & Lane, 2020). Grâce à son faible coût et à sa grande flexibilité, TinyML est particulièrement bien adapté au déploiement d'objets connectés (IoT). Il est utilisé dans une variété d'applications, telles que la reconnaissance vocale sur des assistants personnels (Sohn, Seo & Yoo, 2019), le suivi de données biométriques via des montres connectées (Silva, Monteiro & Oliveira, 2020), ou encore l'analyse de données environnementales à l'aide de capteurs intelligents (Ghahramani *et al.*, 2021).

En combinant efficacité, accessibilité et respect des contraintes des systèmes embarqués, TinyML ouvre la voie à une intégration massive de l'intelligence artificielle dans des dispositifs compacts et économes en énergie (Zhang, Lane *et al.*, 2021).

2.1.1 Autonomie du système

L'un des principaux avantages des systèmes utilisant des modèles TinyML est leur capacité à fonctionner de manière autonome, sans dépendre d'une connexion constante au réseau. Contrairement à certains systèmes intelligents, les modèles TinyML n'ont pas besoin d'être connectés à un serveur distant pour effectuer l'inférence. Toutes les opérations de calcul sont réalisées localement, directement sur le dispositif embarqué (Schizas *et al.* (2022)).

Cette approche présente plusieurs bénéfices majeurs, non seulement elle réduit la latence, mais elle diminue également la consommation d'énergie liée à la transmission de données. Si nécessaire, le dispositif peut transférer uniquement les résultats de l'inférence sur un réseau local ou un serveur distant, réduisant ainsi la quantité de données transmises et optimisant l'utilisation des ressources réseau.

En résumé, un système TinyML bénéficie d'une autonomie bien supérieure à celle d'un système embarqué dépendant d'un serveur distant pour effectuer l'inférence sur un modèle.

2.1.2 Vitesse d'inférence

La vitesse d'exécution des systèmes automatiques devient un critère de plus en plus crucial. Aujourd'hui, ces systèmes sont conçus pour fonctionner en temps réel, c'est-à-dire qu'aucune latence perceptible ne vienne perturber l'obtention des résultats. La vitesse d'inférence, autrement dit le temps nécessaire pour qu'un modèle prédise un résultat, doit donc être aussi faible que possible (Alajlan & Ibrahim (2022)).

L'un des principaux avantages des systèmes embarqués réside dans la simplification des modèles qu'ils exécutent, permettant une rapidité d'exécution accrue, et ce, malgré des capacités de calcul

plus limitées. Ces systèmes sont ainsi optimisés pour allier taille compacte et vitesse élevée, tout en restant compétitifs face à des modèles de grande taille (Ray (2022b)).

De plus, l'exécution locale des modèles offre un avantage déterminant : les prédictions sont effectuées directement sur le dispositif, sans nécessiter l'envoi ou le retour des données vers un centre de calcul distant. Ce temps de transfert, souvent responsable de la majeure partie de la latence, est éliminé. Ainsi, à modèle équivalent, un système embarqué sera toujours plus rapide qu'un système dépendant d'un calcul à distance (Xu *et al.* (2020)).

2.1.3 Consommation énergétique

La consommation énergétique des microcontrôleurs est très faible, ce qui constitue un avantage clé pour les systèmes embarqués. Par exemple, un microcontrôleur nRF5340 consomme environ 8 mA, un ATmega328p autour de 15 mA, et un ESP32-C6 entre 150 et 350 mA, selon la technologie de communication utilisée. Une consommation réduite permet aux systèmes d'être utilisés plus longtemps sans recharge fréquente, améliorant ainsi leur autonomie. De plus, cela évite de recourir à des algorithmes inutilement complexes pour des tâches simples, optimisant ainsi l'efficacité globale du système (Alajlan & Ibrahim (2022)).

2.1.4 Sécurité des données

La sécurité des données constitue une priorité qui ne doit jamais être sous-estimée. Les systèmes embarqués présentent l'avantage de stocker les données directement sur le dispositif lui-même ou sur un serveur/ordinateur local. Cette approche permet de maintenir un contrôle total sur les informations, réduisant ainsi les risques de fuite de données (Xu *et al.* (2020)).

Puisque les données ne transitent pas par des réseaux externes, il devient impossible qu'elles soient utilisées à des fins commerciales. Cette sécurisation revêt une importance capitale, notamment lorsqu'il s'agit de données sensibles telles que des informations biométriques collectées par certains dispositifs de mesure.

Ces données personnelles sont, et doivent rester, privées. Le stockage local apparaît donc comme le moyen le plus efficace pour garantir leur protection. En évitant leur transfert vers des serveurs distants ou des infrastructures en ligne, les risques d'exposition ou d'exploitation malveillante sont considérablement réduits (Schizas *et al.* (2022)).

2.2 Limite des microcontrôleurs

Cependant, les microcontrôleurs ne peuvent pas exécuter tous les types de modèles, car ils sont limités par plusieurs facteurs qui restreignent leur champ d'utilisation. En effet, les modèles les plus récents sont de plus en plus volumineux et nécessitent des technologies avancées d'optimisation, accessibles uniquement sur des systèmes performants (Prieto *et al.* (2016)).

Par exemple, des technologies comme CUDA, conçues pour accélérer considérablement l'exécution des modèles, sont uniquement compatibles avec des systèmes équipés de cartes graphiques NVIDIA. Bien que cela améliore significativement la vitesse des modèles, les microcontrôleurs ne sont pas compatibles avec ce type de solution en raison de leur architecture simplifiée (Chen, Chen & Martin-Kuo (2018a)).

De plus, la majorité des modèles d'intelligence artificielle sont développés dans des langages comme Python, qui ne sont pas directement pris en charge par les microcontrôleurs. Cette incompatibilité impose des contraintes techniques importantes, nécessitant une adaptation préalable des modèles pour qu'ils puissent s'exécuter efficacement sur ces dispositifs à ressources limitées.

2.2.1 Mémoire Flash et SRAM

Le principal problème des microcontrôleurs réside dans leur capacité mémoire limitée. La mémoire embarquée sur ces dispositifs est souvent de l'ordre de quelques centaines de kilo-octets à quelques mégaoctets, avec une mémoire RAM généralement inférieure à 1 Mo. Il est donc nécessaire d'adapter sa stratégie lors du déploiement de modèles sur ces petits appareils (Warden & Situnayake (2019)).

Les microcontrôleurs contiennent deux types de mémoire différentes (Hennessy & Patterson (2011)) : la mémoire Flash, qui stocke le modèle ainsi que le code, et la mémoire RAM, qui permet de manipuler les variables durant l'exécution.

La mémoire Flash est dite non volatile : les données stockées n'ont pas besoin d'être alimentées pour être conservées. Elles restent donc persistantes même si l'alimentation est coupée. La Flash est utilisée pour stocker des données permanentes (firmware et ressources nécessaires au programme), mais elle est plus lente que d'autres types de mémoire, aussi bien en lecture qu'en écriture.

La mémoire RAM, quant à elle, est volatile : les données sont perdues en cas de coupure d'alimentation. Elle stocke les informations temporaires, comme les variables ou les données appelées à changer durant l'exécution. Son avantage réside dans sa rapidité d'accès. Elle contient notamment les variables, les tampons (buffers) et la pile d'exécution.

La RAM, étant la mémoire de travail, doit être suffisamment grande pour contenir l'ensemble du modèle au moment de son exécution. Ainsi, même si un modèle peut tenir dans la mémoire Flash, cela ne garantit pas nécessairement qu'il tiendra dans la RAM.

Tableau 2.1 Comparaison de la mémoire des microcontrôleur communément utilisé pour le TinyML

Microcontroller	Flash	RAM	Consumption (mA)
STM32F103C8 (Cortex-M3)	128	20	30
STM32L476RG (Cortex-M4)	512	128	15
STM32F7746 (Cortex-M7)	1024	320	50
nRF52840 (Cortex-M4)	1024	256	15
ESP32 (Xtensa LX6)	4096	520	150
SAMD21 (Cortex-M0+)	256	32	12
SAMD51 (Cortex-M4)	1024	192	25
PIC32MX (MIPS M4K)	512	128	50
PIC32MZ (MIPS M-Class)	2048	512	80
TM4C123G (Cortex-M4)	256	32	45

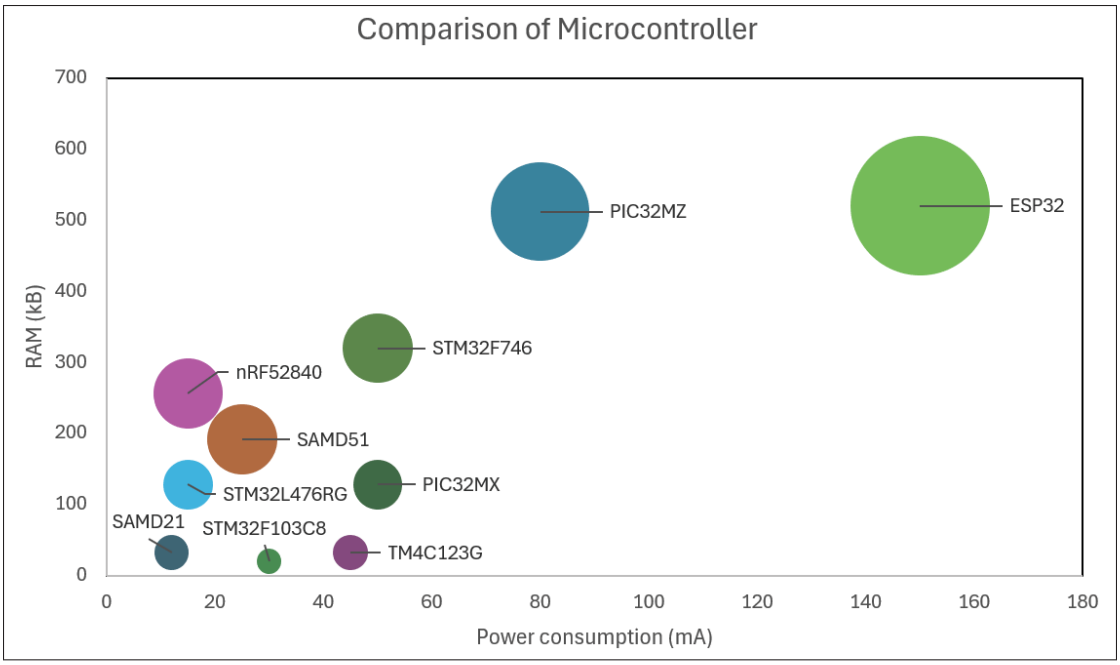


Figure 2.1 Comparaison de la mémoire des microcontrôleur communément utilisé pour le TinyML. La taille des bulles correspond à la capacité de la mémoire Flash

On distingue deux grands types de RAM : la RAM statique (SRAM) et la RAM dynamique (DRAM). La DRAM est largement utilisée dans les ordinateurs, les téléphones et autres appareils nécessitant une grande capacité mémoire. La SRAM, plus rapide et plus stable, se retrouve majoritairement dans les processeurs et les microcontrôleurs, où elle est adaptée aux tâches exigeant des vitesses élevées (par exemple, les caches et la gestion de la pile d'exécution).

Lorsqu'on manipule des données en RAM sur un microcontrôleur, il est essentiel de différencier la pile (stack) du tas (heap). La pile sert à stocker les variables locales, les adresses de retour de fonctions et certains registres. Son espace mémoire peut rapidement être saturé, en particulier lors de la déclaration de tableaux locaux de grande taille, comme c'est parfois le cas dans les applications TinyML. Le tas, quant à lui, est la zone de mémoire allouée dynamiquement : il peut se remplir aisément si l'on y crée de gros blocs de données.

Dans la plupart des architectures de microcontrôleurs, la pile se développe en partant de la fin de l'espace RAM vers le bas, tandis que le tas grandit depuis le début de la RAM vers le haut. Ces

deux zones peuvent donc finir par entrer en collision si l'on ne surveille pas soigneusement leur évolution, provoquant des comportements aléatoires ou même un plantage complet de l'appareil. Il est par conséquent indispensable de contrôler en continu l'évolution de la pile et du tas pour garantir la fiabilité du système déployé sur un microcontrôleur.

2.3 Techniques de réduction de modèle

Pour adapter un modèle existant à une version TinyML, plusieurs techniques permettent de réduire significativement sa taille, ses besoins en puissance de calcul et sa consommation énergétique, tout en minimisant l'impact sur sa précision. Parmi ces techniques, on retrouve l'élagage des poids, qui supprime les paramètres inutiles pour alléger le modèle, la quantification, qui réduit la précision des poids et des activations en passant de 32 bits à 8 bits ou moins, ainsi que le partitionnement, qui divise le modèle en sous-ensembles exécutables indépendamment. Ces méthodes s'accompagnent souvent de processus d'optimisation spécifiques. L'un d'eux est la fusion des opérations, qui consiste à combiner plusieurs étapes du calcul pour réduire le nombre d'opérations nécessaires à l'exécution du modèle. Cette technique permet d'accélérer l'inférence tout en diminuant la consommation énergétique. Par ailleurs, l'adoption de formats adaptés, tels que TensorFlow Lite, ONNX Runtime, ou des bibliothèques optimisées pour les microcontrôleurs comme TensorFlow Lite for Microcontrollers ou TVM, est également courante. Ces optimisations permettent de rendre les modèles compatibles avec les contraintes des microcontrôleurs tout en améliorant leur efficacité énergétique (David *et al.* (2021)).

2.3.1 Élagage des poids

L'élagage des poids est l'une des techniques les plus efficaces et les plus utilisées pour réduire la taille d'un modèle. À l'image de l'élagage des arbres, où l'objectif est de retirer les branches mortes ou celles qui limitent la croissance, l'élagage d'un modèle consiste à supprimer de manière précise les parties les moins significatives de celui-ci. En éliminant soigneusement les poids les moins importants, on réduit le nombre de calculs nécessaires, la mémoire utilisée, et on améliore la vitesse d'inférence du modèle (Hu, Peng, Tai & Tang (2016)).

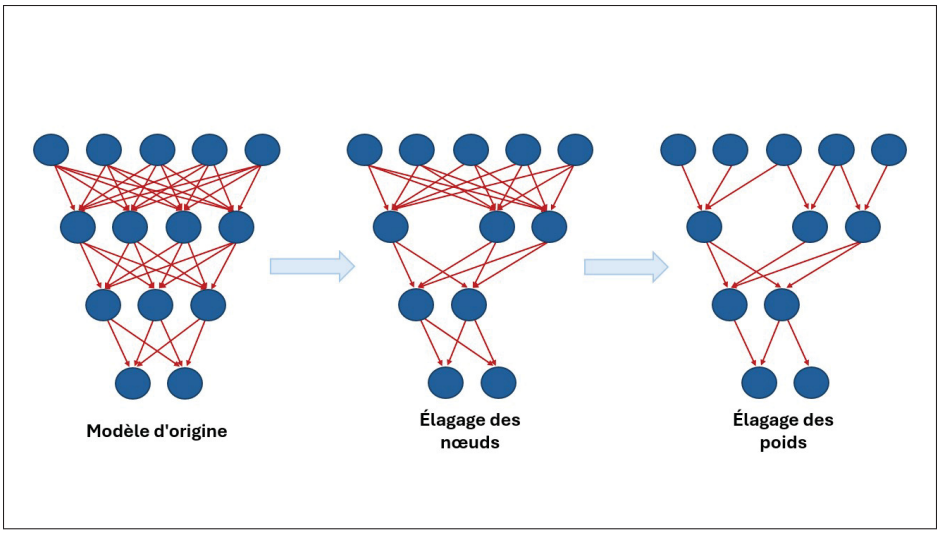


Figure 2.2 Élagage d'un réseau de neurones

Pour effectuer cet élagage, plusieurs méthodes permettent de choisir quels poids peuvent être supprimés. L'une des techniques les plus connues repose sur la régularisation. Cette dernière est basée sur le principe du rasoir d'Ockham, selon lequel les solutions les plus simples doivent être privilégiées. Cela a pour effet de limiter la croissance des poids autour d'une distribution donnée : les valeurs extrêmes, perçues comme un symptôme de sur-apprentissage, sont ainsi pénalisées. Parmi les méthodes de régularisation les plus utilisées figurent la régularisation L1 et L2.

La régularisation L1, appelée LASSO (Least Absolute Shrinkage and Selection Operator), consiste à imposer des contraintes sur les valeurs des poids du modèle. Cette méthode permet à certains poids d'atteindre une valeur nulle, facilitant ainsi l'identification des poids ayant un impact insignifiant sur les prédictions du modèle final. Ces poids peuvent alors être supprimés sans perte significative de précision. La régularisation L1 contribue également à réduire les valeurs extrêmes, limitant ainsi le risque de sur-apprentissage et favorisant une meilleure généralisation. La régularisation L1 utilise la somme des valeurs absolues des résidus, exprimée par la formule suivante :

$$L1(\theta) = \sum_i |\theta_i| + \lambda \cdot \sum_i \mathcal{L}(y_i, \hat{y}_i) \tag{2.1}$$

où θ représente les poids du modèle, λ est un hyperparamètre de régularisation, y_i est la valeur réelle, et $\hat{y}_i$ est la valeur prédite.

La régularisation L2, également appelée ridge ou méthode de Tikhonov, est similaire à L1, mais elle ne permet pas aux poids d'atteindre une valeur nulle. Cette méthode repose sur l'utilisation de la somme des carrés des résidus, donnée par la formule suivante :

$$L2(\theta) = \sum_i \theta_i^2 + \lambda \cdot \sum_i \mathcal{L}(y_i, \hat{y}_i) \tag{2.2}$$

Bien que la régularisation L2 ne facilite pas directement l'identification des poids insignifiants, elle permet néanmoins de réduire la variance du modèle et de limiter les valeurs extrêmes, contribuant ainsi à une meilleure généralisation.

Une fois les poids mieux distribués grâce à la régularisation, il devient possible de définir un seuil en dessous duquel les poids peuvent être supprimés. Pour obtenir une réduction précise, ce seuil peut être ajusté dynamiquement afin d'atteindre un pourcentage de réduction donné. Par exemple, pour réduire le nombre de poids de 50%, on peut augmenter progressivement le seuil jusqu'à ce que 50% des poids du modèle soient supprimés. Ce processus d'élagage permet une optimisation progressive et contrôlée de la taille du modèle.

Cependant, l'élagage nécessite une phase de ré-entraînement pour ajuster à nouveau les poids restants et compenser les pertes potentielles liées à la suppression des poids. Cette étape est nécessaire pour préserver les performances du modèle après la réduction.

En fonction de la complexité et du type de modèle, le processus d'élagage peut réduire la taille d'un réseau de 50 à 95%, sans entraîner de perte significative en termes de précision. Ce gain d'efficacité est particulièrement précieux pour le déploiement sur des appareils à ressources

limitées. Toutefois, il convient de noter que cette méthode exige une attention particulière : un élagage excessif peut nuire à la précision et à la robustesse du modèle.

2.3.2 Quantification des poids

La quantification des poids est une méthode très efficace pour réduire l'empreinte mémoire d'un modèle. Elle consiste à diminuer la précision de chaque poids en utilisant un type de variable plus compact. Le plus souvent, on convertit les variables en format float32 (variable flottante sur 32 bits) en int8 (entier sur 8 bits). Cette conversion permet de diviser par quatre la mémoire nécessaire pour stocker les poids du modèle. C'est donc une technique simple à mettre en place, qui réduit de manière significative l'espace mémoire requis pour faire tourner le modèle (Krishnamoorthi (2018)). Cette méthode est particulièrement efficace pour les modèles comprenant un grand nombre de poids, car une légère perte de précision sur certains poids impacte très peu la performance globale de la prédiction. En effet, il a été démontré que les poids peuvent s'influencer mutuellement pour obtenir une précision finale acceptable, même si la précision individuelle de chaque poids est réduite. Un autre avantage de cette méthode est qu'elle simplifie les calculs. Les opérations en virgule flottante (float) sont parfois coûteuses en puissance de calcul. En limitant les calculs aux entiers, on peut améliorer la vitesse d'inférence du modèle. La quantification est souvent utilisée pour des modèles ayant été préalablement entraînés avec une haute précision, mais qui nécessitent ensuite une réduction de taille pour les applications. C'est notamment le cas pour les modèles de type LLM (Large Language Models) : ceux-ci sont entraînés en haute précision, mais peuvent être quantifiés pour réduire les coûts et la latence d'inférence pour les utilisateurs finaux. Par exemple, GPT-2 est disponible en plusieurs précisions, et des applications de ce modèle utilisent la quantification pour minimiser le coût de l'inférence.

La quantification peut être réalisée de deux façons différentes. La première méthode, appelée quantification après entraînement, est la plus simple à mettre en œuvre, car elle intervient après l'entraînement du modèle. Cependant, cette approche entraîne généralement une légère perte de précision, typiquement de l'ordre de 2 à 3%. Pour effectuer cette opération, deux paramètres sont

nécessaires : L'échelle (scale), qui représente le facteur de mise à l'échelle utilisé pour convertir les valeurs flottantes en valeurs entières. Le point zéro (zero-point), un décalage appliqué pour aligner les valeurs quantifiées avec l'échelle souhaitée.

La formule générale pour retrouver la valeur de nos poids d'origine est la suivante :

$$x = S \times (x_q - Z) \tag{2.3}$$

où S représente l'échelle, les poids quantifiés, et Z le point zéro.

Pour obtenir les poids quantifiés, on utilise la formule suivante :

$$x_q = \text{round}\left(\frac{x}{S} + Z\right) \tag{2.4}$$

L'échelle S est généralement définie comme :

$$s = \frac{\text{max\_float} - \text{min\_float}}{\text{max\_int} - \text{min\_int}} \tag{2.5}$$

où les valeurs float correspondent aux valeurs extrêmes des poids d'origine, et les valeurs int correspondent aux limites de la plage entière souhaitée, qui est généralement [-128,127] pour des entiers sur 8 bits (int8).

Le point zéro Z est ensuite calculé par la formule :

$$Z = \text{round}\left(-\frac{\text{min\_float}}{s}\right) \tag{2.6}$$

La quantification des poids est appliquée de manière indépendante à chaque couche, permettant ainsi d'assigner une échelle et un point zéro uniques à chaque couche du modèle. En effet, chaque couche d'un modèle a une distribution différente de valeurs ; il est donc essentiel de quantifier chaque couche séparément pour conserver autant que possible la précision du modèle d'origine.

Cette approche réduit les erreurs dues à la quantification et permet d'obtenir un modèle quantifié dont les performances sont très proches de celles du modèle de base.

Une seconde méthode de quantification, un peu plus complexe à mettre en œuvre, est la quantification durant l'entraînement, ou quantization-aware training (QAT). Cette méthode permet de quantifier le modèle directement pendant l'entraînement, sans nécessiter de transformation supplémentaire après l'entraînement. Contrairement à la quantification après entraînement, qui peut entraîner une légère perte de précision, la précision obtenue avec la quantification durant l'entraînement est la précision finale du modèle. Pour ce faire, le modèle est entraîné avec une précision élevée, mais il simule les effets de la quantification des poids et des activations en appliquant une quantification périodique tout au long de l'entraînement. Cette approche permet au modèle de compenser progressivement les erreurs introduites par la quantification. En mélangeant des valeurs flottantes et entières pendant l'entraînement, on maintient un ajustement précis des poids, car les gradients sont toujours calculés en virgule flottante. La quantification durant l'entraînement est donc la méthode la plus adaptée aux environnements contraints, tels que les systèmes embarqués, où la mémoire et la puissance de calcul sont limitées.

2.3.3 Fusion des opérations

La fusion des opérations est une méthode qui permet de réduire le nombre de calculs à effectuer en combinant une suite de calculs en une seule étape. Elle permet de réduire la latence et la consommation de mémoire des systèmes embarqués. La combinaison des opérations mathématiques permet de réduire les appels mémoire, ce qui optimise l'utilisation du cache et la charge computationnelle (Boehm, Reinwald, Hutchison, Evfimievski & Sen (2018)).

2.3.4 Partitionnement du modèle

Le partitionnement de modèle peut être entrepris pour diviser un modèle volumineux en plusieurs segments plus petits. Dans le cas des modèles embarqués, cette approche est très utile car elle permet d'utiliser des modèles trop grands pour tenir dans la RAM d'un microcontrôleur, mais

suffisamment petits pour être stockés dans la mémoire flash. La segmentation permet donc de découper un modèle en plusieurs parties, chacune suffisamment petite pour être chargée individuellement dans la RAM. L'inférence du modèle se fait alors par étapes : la première partition est chargée dans la RAM pour être exécutée, puis la sortie de cette partition est conservée en mémoire. Ensuite, la partition suivante est chargée en mémoire, en utilisant les valeurs de sortie de la partition précédente comme entrée. Ce processus est répété pour traverser toutes les partitions du modèle. Cette méthode est très efficace pour faire tourner des modèles bien plus volumineux que ce que la RAM peut normalement accueillir (Lim, Lee, Kwak & Kim (2024)). Cependant, ce chargement et déchargement répété des partitions ralentit le modèle, car chaque étape d'inférence nécessite un transfert de données entre la mémoire flash et la RAM. Il est néanmoins à noter que la quantité de données transférées reste relativement faible dans le cas d'un microcontrôleur, ce qui limite l'impact sur la performance globale.

Le partitionnement est également largement utilisé pour les modèles de très grande envergure. Ces modèles, qui peuvent atteindre des milliards de paramètres, sont tout simplement trop volumineux pour être entraînés et exécutés sur des ordinateurs classiques. Pour ce faire, le modèle est segmenté et les différentes parties sont entraînées en parallèle sur plusieurs ordinateurs. Cette approche permet d'envisager la création de modèles sans limite de taille, pouvant être partitionnés et distribués sur plusieurs machines. Cependant, cette technique présente des défis techniques considérables, car les données doivent être séparées et réassemblées de manière répétée, ce qui entraîne des contraintes de synchronisation, de communication et de coordination. Ces aspects ajoutent une complexité significative au processus de formation et d'inférence, nécessitant des infrastructures spécifiques pour gérer la répartition et la communication entre les machines.

2.4 Compilation C

Une fois le modèle entraîné et optimisé via des frameworks de machine learning, il doit être compilé dans un format compatible avec le microcontrôleur ciblé. En effet, les modèles d'apprentissage sont généralement développés en Python, un langage qui n'est pas pris en

charge par la plupart des microcontrôleurs. De plus, les opérations couramment utilisées, comme la convolution ou la normalisation, ne peuvent pas être exécutées directement par un microcontrôleur.

Cette étape de compilation est donc essentielle pour traduire les opérations du modèle en instructions optimisées pour la cible, par exemple un microcontrôleur ARM Cortex-M4 (Sponner, Waschneck & Kumar (2021); Liu *et al.* (2023)).

La compilation en langage C permet une gestion de bas niveau de l'architecture du microcontrôleur, offrant ainsi un contrôle total du matériel. Le code résultant de cette compilation est dépourvu de toute donnée superflue. Les blocs de données sont optimisés pour la mémoire de la cible, les buffers sont adaptés à la taille et aux besoins du modèle, et le code est optimisé pour une latence minimale, garantissant des performances efficaces même dans des environnements contraints en ressources.

2.5 Framework orienté pour de l'embarqué

Pour compiler notre modèle et exécuter des inférences, il est nécessaire d'utiliser des outils capables de le convertir dans un langage compatible avec la cible. Plusieurs frameworks permettent cette conversion, tels que TensorFlow Lite Micro, qui facilite le transfert des modèles développés sous TensorFlow vers un format allégé adapté aux microcontrôleurs (Warden & Situnayake (2019)). Un autre outil, MicroTVM, a été développé pour être compatible avec une large variété de frameworks de machine learning, offrant ainsi une plus grande flexibilité dans le déploiement des modèles sur des architectures embarquées (Liu *et al.* (2023)).

2.5.1 Tensorflow Lite micro

TensorFlow est un framework puissant pour le développement de modèles d'apprentissage machine performants. En plus de ses fonctionnalités avancées, il propose une version allégée, TensorFlow Lite (TFLite), qui permet un meilleur contrôle sur la taille des modèles, répondant ainsi aux exigences des systèmes aux contraintes plus strictes, comme les dispositifs embarqués.

Par ailleurs, TensorFlow Lite offre un outil dédié, TensorFlow Lite Micro (TFLite Micro), conçu pour convertir ces modèles allégés en versions optimisées pour les microcontrôleurs (Warden & Situnayake (2019)).

TFLite Micro est un outil extrêmement efficace qui simplifie considérablement le déploiement des modèles sur des microcontrôleurs. Il permet une transition fluide d'un modèle développé avec TensorFlow vers une version compilée en C, directement exploitable sur des plateformes embarquées. En outre, TFLite Micro intègre des mécanismes pour appliquer automatiquement des techniques de réduction de taille de modèle, telles que la quantisation (réduction de la précision des poids) et l'élagage des poids (pruning), afin d'optimiser les performances sur des dispositifs aux ressources limitées.

Cependant, malgré sa simplicité d'utilisation et son efficacité, TFLite Micro offre un contrôle limité sur l'optimisation fine des modèles résultants. Il ne permet pas de personnaliser précisément les optimisations en fonction des spécificités de chaque microcontrôleur ou des exigences de l'application cible. Par conséquent, pour des optimisations avancées et sur mesure, il devient nécessaire de le combiner avec d'autres frameworks, comme MicroTVM, qui permettent un contrôle beaucoup plus précis et flexible des ajustements du modèle.

2.5.2 MicroTVM

TVM est un framework open source dédié à l'optimisation des modèles de deep learning. Combiné à des frameworks tels que TensorFlow, PyTorch ou ONNX, il offre un meilleur contrôle sur l'ajustement et l'optimisation des modèles, en tenant compte des spécificités des plateformes cibles. MicroTVM est une extension de TVM, spécialisée dans l'optimisation pour les dispositifs embarqués, tels que les microcontrôleurs, notamment ceux basés sur l'architecture ARM Cortex-M4 (Liu *et al.* (2023)).

L'un des principaux atouts de MicroTVM est sa capacité à proposer des optimisations spécifiques aux contraintes des plateformes embarquées, contrairement aux approches généralistes. Une de ses forces réside dans la gestion avancée de la mémoire, essentielle pour minimiser la latence et

permettre une inférence rapide sur des microcontrôleurs aux ressources limitées. Pour atteindre cet objectif, MicroTVM utilise des méthodes de fusion d'opérations adaptées au contrôleur ciblé. En outre, il offre un partitionnement précis des modèles en fonction de la taille de chaque couche et de l'espace mémoire disponible, optimisant ainsi l'utilisation de la SRAM et de la mémoire flash.

Un autre avantage majeur de MicroTVM réside dans ses outils d'optimisation automatique, notamment AutoTVM. Lorsqu'AutoTVM est appliqué à un microcontrôleur, il simule l'inférence directement sur la cible matérielle. Cela permet d'identifier les paramètres les plus adaptés pour maximiser les performances du modèle tout en tenant compte des contraintes spécifiques du matériel. Grâce à cette approche, il est possible de contrôler le compromis entre précision et rapidité d'inférence, ce qui est particulièrement crucial dans les applications embarquées.

MicroTVM est conçu pour offrir des inférences parmi les plus rapides, mais cela peut parfois se traduire par une utilisation légèrement plus élevée de la mémoire par rapport à des outils d'optimisation plus généralistes. Cependant, sa flexibilité et ses capacités d'optimisation spécifiques en font un leader dans le domaine des microcontrôleurs. Cette combinaison d'efficacité et de personnalisation en fait un outil privilégié pour les développeurs cherchant à exploiter pleinement le potentiel de leurs dispositifs embarqués.

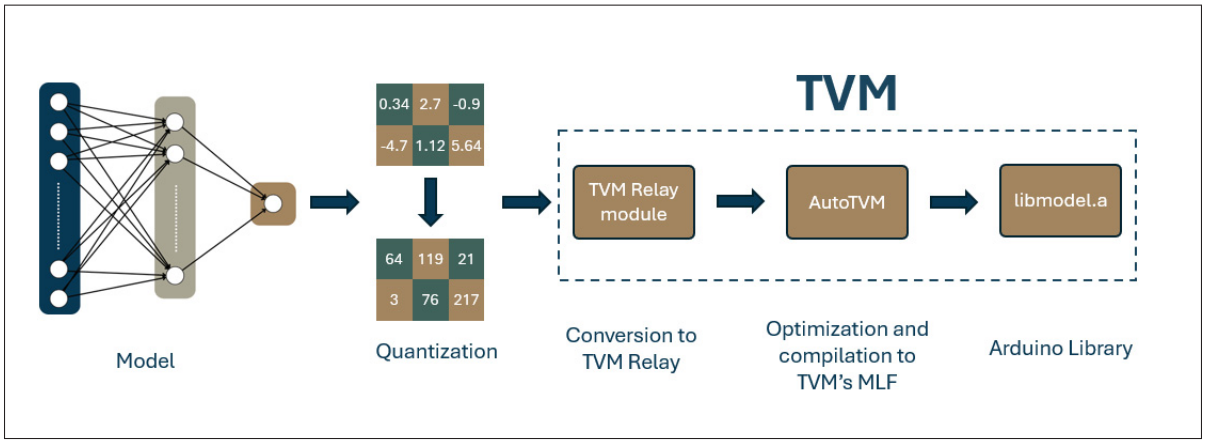


Figure 2.3 Pipeline pour la mise en place de microTVM

CHAPITRE 3

ÉLECTRONIQUE IMPRIMABLE

Les circuits imprimés classiques, ou PCB, reposent sur des techniques soustractives consistant à retirer de la matière sur une ou plusieurs feuilles de cuivre laminé sur un substrat de FR4 afin de dessiner le circuit désiré. Sur un autre pied, l'électronique imprimable adopte une approche additive : au lieu d'enlever de la matière, on dépose de l'encre conductrice (à base d'argent, de cuivre, d'or, etc.) uniquement aux endroits nécessaires. Cette méthode permet non seulement de réaliser des circuits sur une grande variété de supports (tels que les plastiques, textiles, ou films polymères), mais offre également la possibilité de fabriquer des circuits flexibles. Chaque type de substrat présente cependant des contraintes spécifiques, notamment en termes d'adhérence, de résistance thermique et de flexibilité.

Sur le plan technique, cette méthode permet de réduire considérablement le gaspillage de matière et d'éliminer plusieurs étapes de traitement impliquant des produits chimiques agressifs, inhérents à la fabrication traditionnelle des PCBs. D'un point de vue économique et écologique, l'optimisation de la dépose de matière se traduit par une diminution des coûts de production et un impact environnemental réduit, faisant de l'électronique imprimable une alternative prometteuse aux procédés conventionnels. Cette technologie ouvre ainsi de nouvelles perspectives dans la conception de dispositifs électroniques innovants, flexibles et intégrables dans des environnements variés.

3.1 Encre conductrice

Pour réaliser ces circuits, des encres conductrices sont utilisées. Celles-ci sont formulées à partir de particules métalliques liées à un polymère permettant son impression. La composition de ces encres repose sur la concentration en particules métalliques pour garantir une conductivité élevée tout en maintenant une viscosité compatible avec les procédés d'impression. Un des principaux défis techniques consiste à assurer une dispersion homogène des particules afin d'éviter leur agglomération, provoquant des soucis au niveau de l'impression ainsi que sur le

comportement du circuit (Picard (2018)). Une fois imprimé, un procédé de frittage est effectué pour venir fusionner les particules métalliques entre elles et obtenir un film conducteur. Ce traitement peut être réalisé par chauffage thermique, par lampes flash ou même par techniques de frittage laser. Chacune de ces techniques permet d'offrir des avantages spécifiques en termes de rapidité, d'efficacité énergétique et de compatibilité avec divers substrats, notamment ceux sensibles à la chaleur, comme le papier (Owens *et al.* (2021)).

3.2 Encre semi-conductrice

Les encres semi-conductrices représentent une avancée significative dans le domaine de l'électronique imprimable. Contrairement aux encres conductrices, elles intègrent des matériaux aux propriétés semi-conductrices, tels que des oxydes métalliques, des polymères organiques ou des nanoparticules fonctionnalisées. Ces formulations permettent de réaliser des dispositifs actifs permettant la fabrication de capteurs imprimés capables de détecter des variations de pression, de température, d'humidité ou de composition gazeuse. Une autre application prometteuse des encres semi-conductrices réside dans la fabrication d'électrodes conductrices transparentes (TCE) pour des dispositifs optoélectroniques flexibles. Ces technologies ouvrent également des perspectives dans le domaine photonique, notamment pour des dispositifs d'éclairage avancés, des capteurs optiques de nouvelle génération ou encore des cellules photovoltaïques imprimées (Gerlein, Benavides-Guerrero & Cloutier (2024)).

3.3 Capteurs imprimés

Les capteurs imprimés tirent pleinement parti des encres semi-conductrices, permettant la réalisation de dispositifs souples, légers et à faible coût. Ces capteurs fonctionnent grâce à des propriétés électriques qui varient en réponse à des stimuli externes tels que la pression, la température, l'humidité ou la présence de gaz. Par exemple, un capteur de pression imprimé peut être conçu de manière à ce que la résistance électrique du matériau change lorsqu'une force est appliquée, facilitant ainsi la conversion de la contrainte mécanique en signal électrique. Ces capteurs sont devenus indispensables pour le monitoring et l'analyse des mouvements

biomécaniques. Intégrés à même les semelles, ils peuvent également faire un suivi en temps réel de la posture et de l'équilibre des personnes.

De même, les capteurs de température et d'humidité imprimés permettent de suivre précisément les conditions environnementales, et permettent même d'obtenir des plages de sensibilité plus grandes et flexibles grâce au développement des encres céramiques résistantes à de hautes températures (Fourmont, Vaussenat, Gratuze, Ross & Cloutier (2024); Wadhwa, Benavides-Guerrero, Gratuze, Bolduc & Cloutier (2024)).

Les avantages des capteurs imprimés incluent leur flexibilité, leur intégrabilité sur des surfaces non conventionnelles et la possibilité d'une production en grande série via des procédés d'impression. Ces caractéristiques ouvrent des perspectives intéressantes dans des domaines tels que l'internet des objets, la santé connectée et les dispositifs portables.

3.4 Impression par sérigraphie

Pour réaliser ces capteurs, plusieurs méthodes peuvent être mises en place, cependant la méthode d'impression par sérigraphie est largement répandue dans divers domaines. Traditionnellement utilisée pour la confection de motifs sur textile (par exemple pour la fabrication de vêtements), cette technique s'est imposée comme l'une des principales méthodes pour la production d'électronique imprimée. La sérigraphie est une technique d'impression qui utilise un écran (ou pochoir) pour transférer l'encre sur le substrat de manière localisée. Le processus se déroule en plusieurs étapes clés :

- **Préparation de l'écran :** Un écran est enduit d'un matériau photosensible, puis exposé à la lumière à travers un film contenant le motif désiré. Les zones non exposées sont ensuite retirées, créant ainsi le pochoir.
- **Dépôt de l'encre :** L'encre, qui peut être conductrice, semi-conductrice ou isolante selon l'application, est déposée sur l'écran. Une raclette permet de forcer l'encre à travers les zones ouvertes du pochoir pour la transférer sur le substrat.

- **Séchage et frittage :** Après l'impression, le substrat subit un traitement thermique ou un processus de frittage afin de consolider l'encre, d'améliorer son adhérence et d'optimiser ses propriétés électriques.

Ce procédé est particulièrement adapté pour la production de circuits flexibles et d'électronique sur des substrats variés tels que le textile, le plastique ou le papier. Il est préféré pour obtenir des couches plus épaisses de matériaux. Toutefois, la résolution des motifs est limitée par la finesse du maillage de l'écran et par la viscosité de l'encre.

CHAPITRE 4

REVUE DE LITTÉRATURE

L'émergence de capteurs imprimés à faible coût ouvre de nouvelles perspectives dans le domaine biomédical, notamment grâce à l'intégration de nanomatériaux et à la fonctionnalité offerte par l'apprentissage automatique. Ces capteurs offrent l'avantage d'être flexibles, miniaturisés et adaptés à une fabrication à grande échelle, ce qui est particulièrement intéressant pour la surveillance continue et en temps réel des paramètres physiologiques.

4.1 Capteurs imprimés pour le domaine biomédical

Les récentes avancées dans les techniques d'impression ont permis le développement de capteurs souples, miniaturisés et à faible coût, adaptés à une production industrielle de grande échelle. Ces dispositifs, réalisés sur des substrats flexibles tels que le papier, se révèlent particulièrement adaptés aux exigences du domaine biomédical, en offrant des solutions portables et non invasives.

Une des applications de ces technologies concerne la surveillance respiratoire. Dans l'étude de Maier *et al.* (2019), un capteur en papier a été développé pour mesurer en temps réel l'hydrogène peroxyde présent dans l'haleine. Ce biomarqueur, indicateur du stress oxydatif, offre des pistes intéressantes pour le suivi de pathologies respiratoires ou métaboliques. De même, l'article de Lu, Haider, Gardner, Alexander & Massoud (2019) présente un capteur à base de graphène imprimé sur papier, capable de détecter avec une grande sensibilité les variations du débit respiratoire. La haute sensibilité du graphène permet ainsi de faciliter le diagnostic de troubles tels que l'asthme ou la bronchopneumopathie chronique obstructive (BPCO).

En parallèle, l'article de Bariya *et al.* (2018) démontre que les procédés industriels, tels que la gravure en rouleau, permettent de fabriquer des capteurs électrochimiques performants et économiques. Ces capteurs sont capables de détecter une large gamme d'analytes, allant des biomarqueurs chimiques aux paramètres physiologiques, et s'intègrent parfaitement dans des dispositifs portables destinés à une surveillance médicale en temps réel.

Outre ces approches classiques, l'intégration de nanomatériaux offre de nouvelles perspectives pour améliorer la sensibilité et la sélectivité des capteurs imprimés. Par exemple, Pandit, Banerjee, Srivastava, Nie & Pan (2019) ont mis en évidence l'utilisation innovante de points quantiques de carbone (carbon dots) fonctionnalisés pour la détection d'analytes spécifiques. La fluorescence modulée de ces nanostructures en présence d'une cible permet une quantification très précise du biomarqueur concerné. Lorsqu'elles sont associées à des modèles d'apprentissage automatique, (tels que les machines à vecteurs de support (SVM), le K-NN ou les arbres de gradient boosté (GBT), ces capteurs affichent des taux de prédiction atteignant les 100%, ouvrant la voie à des diagnostics rapides et fiables dans des dispositifs portables ou connectés via l'IoT.

Dans une démarche similaire, Behera *et al.* (2021) ont exploité les propriétés des nano-feuilles de MoS_2 pour la détection de protéines dans le sérum sanguin. La conjugaison de ces matériaux avec des marqueurs fluorescents, combinée à l'application d'algorithmes tels que l'analyse discriminante linéaire (LDA) et les réseaux de neurones, a permis d'atteindre des précisions supérieures à 95%. Ces résultats illustrent le potentiel des capteurs imprimés non seulement pour réduire les coûts et le temps d'analyse, mais aussi pour offrir une qualité diagnostique équivalente voire supérieure à celle des méthodes conventionnelles.

D'autres recherches, comme celles de Chen, He & Cheng (2020), mettent en avant des capteurs peu intrusifs fabriqués à partir de nanotubes de carbone à simple feuillet pour le diagnostic de la respiration. En exploitant des modèles de convolution et des analyses temporelles, ces dispositifs classifient efficacement divers composés présents dans l'haleine. Par ailleurs, les études sur des capteurs à base de nanofils, présentées par Khan, Thomson, Debnath, Motayed & Rao (2020b) dans le domaine de la détection de gaz, suggèrent que des approches analogues pourraient être adaptées au suivi de paramètres biomédicaux, renforçant ainsi le pont entre technologies environnementales et médicales.

L'intégration de ces capteurs innovants dans des réseaux intelligents permet de relever des défis majeurs liés à la collecte et à l'interprétation des données. Des travaux comme ceux d'Ortega, González-Prieto, Bobadilla & Gutiérrez (2020) et de Pan & Li (2010) sur la prédiction de

valeurs manquantes dans les réseaux de capteurs mettent en lumière l'importance de développer des algorithmes robustes garantissant la fiabilité des mesures dans des environnements réels. Cette convergence technologique, alliant techniques d'impression, nanomatériaux et stratégies d'apprentissage automatique, trace la voie vers la réalisation de systèmes de santé connectée. Ces systèmes sont capables de fournir des diagnostics en temps réel tout en s'adaptant aux contraintes de miniaturisation et de faible consommation énergétique, ce qui représente un atout considérable pour l'avenir de la médecine personnalisée.

En somme, l'évolution des technologies d'impression, enrichie par l'incorporation de nanomatériaux et la synergie avec des algorithmes d'apprentissage, ouvre de nouvelles perspectives dans le développement de capteurs biomédicaux. Ces dispositifs, qu'ils soient destinés à la surveillance de la respiration ou à la détection précise de biomarqueurs, contribuent à des diagnostics précoces et à une meilleure gestion des pathologies, renforçant ainsi l'essor des systèmes de santé connectée.

4.2 Utilisation d'algorithmes d'apprentissage pour le médical

L'intégration des algorithmes d'apprentissage automatique dans les dispositifs médicaux révolutionne l'analyse des données issues de capteurs, permettant ainsi des diagnostics plus précis, rapides et en temps réel. En combinant des modèles sophistiqués avec des capteurs innovants (notamment ceux issus des techniques d'impression abordées précédemment) il est désormais possible de détecter, classifier et interpréter une grande variété de signaux physiologiques, tout en tenant compte des contraintes des dispositifs embarqués.

Plusieurs approches algorithmiques se distinguent dans le traitement des données biomédicales :

- **Réseaux de neurones convolutifs (CNN)** : Ces modèles sont particulièrement efficaces pour l'analyse d'images et de signaux complexes. Par exemple, les travaux de Münzner *et al.* (2017) ont démontré l'efficacité de la fusion de données multimodales dans la reconnaissance d'activités humaines, une approche qui ouvre des perspectives intéressantes pour la surveillance des patients en rééducation ou la détection précoce de troubles moteurs.

- **Machines à vecteurs de support (SVM) et régressions** : Utilisés pour classifier et interpréter des signaux physiologiques, ces algorithmes optimisent l'analyse des variations captées par des capteurs, facilitant ainsi la détection de tendances anormales ou la prédiction de crises chez des patients atteints de pathologies respiratoires chroniques.
- **Techniques d'optimisation et d'extraction de caractéristiques** : L'association d'outils comme l'analyse en composantes principales (PCA) avec des SVM, illustrée par Zhou *et al.* (2020) dans la traduction du langage des signes via des réseaux de capteurs extensibles, démontre l'étendue des applications possibles. De plus, l'optimisation des modèles (par exemple via la mitigation des problèmes de gradient Tan & Lim (2019) ou l'optimisation de la régression logistique Zou, Hu, Tian & Shen (2019)) assure des performances élevées même sur des plateformes à ressources limitées.

L'apprentissage automatique ne se contente pas d'améliorer la précision des diagnostics, il contribue également à rendre les systèmes de surveillance médicale plus adaptatifs et fiables :

- **Estimation de données manquantes et filtrage collaboratif** : L'utilisation d'algorithmes tels que le K-NN, notamment dans l'estimation de données manquantes (Pan & Li (2010)), ou des techniques de filtrage collaboratif pour prédire les valeurs dans des réseaux étendus (Ortega *et al.* (2020)), permettent d'améliorer la robustesse des systèmes de collecte de données.
- **Personnalisation des diagnostics** : En analysant continuellement les données collectées par des capteurs intelligents, il devient possible de personnaliser les diagnostics et d'adapter les traitements aux profils individuels des patients, ouvrant ainsi la voie à une médecine plus réactive et personnalisée.

En résumé, l'intégration de l'apprentissage automatique dans le domaine médical, couplée aux avancées en capteurs imprimés et en nanotechnologies, transforment en profondeur la manière dont les données de santé sont collectées et analysées. Le mariage entre des modèles d'IA performants (tels que les CNN, SVM et autres techniques d'optimisation) et des dispositifs à

faible consommation (grâce à TinyML) offre des solutions innovantes pour des diagnostics précoces, une gestion en temps réel des pathologies, et, à terme, une meilleure qualité de vie pour les patients. Les défis technologiques actuels, notamment en termes d'optimisation énergétique et de robustesse des modèles, ouvrent également des perspectives de recherche passionnantes pour l'avenir des systèmes de santé connectée.

4.3 État de l'art de l'IA embarquée

L'intelligence artificielle embarquée est un domaine qui bénéficie d'optimisations issues de l'ensemble du spectre de l'intelligence artificielle. Il est désormais de plus en plus facile de convertir des modèles autrefois jugés impossibles à intégrer dans un microcontrôleur. Cette évolution permet au TinyML de tirer parti des améliorations dédiées aux modèles de machine learning et de deep learning plus complexes. Cependant, réduire la taille des modèles ne suffit pas : les nombreuses contraintes des systèmes embarqués nécessitent des adaptations spécifiques. La recherche tend également à simplifier et à automatiser la conversion des modèles traditionnels en versions adaptées au TinyML, favorisant ainsi leur déploiement pour des tâches spécifiques.

Comme le souligne Ray (2022a), l'évolution du TinyML ne peut se faire sans une adaptation simultanée du matériel. On observe aujourd'hui des microcontrôleurs, DSPs (Digital Signal Processors) et ASICs (Application-Specific Integrated Circuits) optimisés pour l'exécution de modèles de machine learning. Ces systèmes intègrent des unités de calcul spécialisées, telles que les NPU (Neural Processing Units), conçues pour l'inférence de réseaux de neurones et les calculs matriciels. De plus, ces dispositifs sont dotés de mémoires améliorées, plus grandes et architecturées différemment pour optimiser la gestion des caches. Enfin, ils intègrent également des moteurs cryptographiques pour protéger les modèles d'IA et les données, renforçant ainsi la sécurité dans des applications sensibles, notamment dans le domaine médical.

Tous les microcontrôleurs adaptés à l'IA sont aujourd'hui développés en tenant compte des différents frameworks tels que TensorFlow Lite Micro ou TVM, afin d'optimiser la compatibilité et la vitesse d'inférence. Il devient donc de plus en plus difficile de comparer ces nouvelles

combinaisons de processeurs et d'optimisations. Une nouvelle méthodologie se met en place pour mesurer la performance de ces dispositifs en termes d'inférence, de consommation énergétique et d'empreinte mémoire. Il devient essentiel de disposer de références concrètes pour choisir la plateforme la plus adaptée au développement précis de chaque application.

L'équipe de Osman, Abid, Gemma, Perotto & Brunelli (2021) a comparé deux frameworks TinyML : TensorFlow Lite for Microcontrollers et STM32Cube.AI, respectivement sur l'Arduino Nano 33 BLE Sense (nRF52840) et le STM32 NUCLEO-F401RE (LQFP64), afin d'établir des critères de sélection des frameworks TinyML en fonction des besoins applicatifs. Les auteurs ont testé trois modèles de classification basés sur des réseaux de neurones : le TinyMLPerf Model, le Speech Commands Model et le Person Detection Model. Les performances ont été évaluées selon plusieurs critères : la taille mémoire (Flash et RAM), le temps d'inférence (latence) et la consommation énergétique.

Cette évaluation a montré que des frameworks comme TensorFlow Lite for Microcontrollers sont très performants et flexibles, mais au prix d'une consommation mémoire et d'une latence plus élevées. À l'inverse, des frameworks comme STM32Cube.AI se révèlent particulièrement efficaces pour les microcontrôleurs de la famille STM32, offrant des performances optimisées en termes de rapidité et de consommation. Cependant, cette efficacité peut être limitée par la compatibilité restreinte avec d'autres plateformes.

Ainsi, bien qu'un microcontrôleur puisse sembler plus performant, il peut être contraint par le framework utilisé. Un framework plus généraliste permettra une adaptation plus facile du modèle à différentes plateformes, mais au détriment d'une optimisation fine. En revanche, des outils offrant un contrôle en profondeur des processus des microcontrôleurs peuvent limiter la portabilité des modèles entre différentes plateformes.

C'est pour résoudre ces problèmes que des plateformes comme TVM, avec son outil AutoTVM, permettent à la fois de mettre en place un framework flexible et compatible avec un large éventail de microcontrôleurs, tout en offrant une optimisation en profondeur. AutoTVM permet d'ajuster

les performances des microcontrôleurs en fonction des besoins spécifiques de l'application, qu'il s'agisse de prioriser la vitesse, la consommation énergétique ou l'empreinte mémoire.

AutoTVM repose sur une approche d'auto-tuning : il teste différentes stratégies d'optimisation et compare leur impact sur la latence et la consommation mémoire. Cette méthode est particulièrement utile pour trouver un équilibre entre la vitesse d'inférence et l'efficacité énergétique. De plus, AutoTVM s'adapte efficacement à la cible en exploitant des accélérateurs spécifiques, comme les DSP intégrés aux Cortex-M4F.

En complément de TensorFlow Lite for Microcontrollers, AutoTVM corrige l'aspect parfois trop généraliste de ce framework en appliquant des optimisations spécifiques à la cible matérielle et à l'application. Cela permet d'améliorer significativement les performances des modèles sur des systèmes embarqués contraints, comme le souligne Immonen & Hämäläinen (2022).

4.3.1 Approches complémentaires et nouvelles perspectives

Afin de relever les défis liés à l'exécution de modèles sur des systèmes contraints, plusieurs approches complémentaires ont émergé :

- **MicroNets pour microcontrôleurs** L'article de Banbury *et al.* (2021) propose des architectures de réseaux de neurones spécialement conçues pour les microcontrôleurs courants. En réduisant le nombre de paramètres et en simplifiant la structure des modèles, ces architectures permettent de maintenir des performances satisfaisantes tout en minimisant la consommation de ressources, illustrant ainsi la tendance à repenser le deep learning pour des environnements extrêmement contraints.
- **MATCH : Compilation model-aware pour dispositifs hétérogènes** La plateforme MATCH de Hamdi *et al.* (2024) étend les capacités de TVM en introduisant une phase de compilation "model-aware". Cette approche ajuste automatiquement les optimisations en fonction des caractéristiques spécifiques du modèle et de l'architecture hôte, permettant ainsi de tirer le meilleur parti de dispositifs hétérogènes et de trouver un équilibre optimal entre performance, consommation énergétique et empreinte mémoire.

- **RELAX : Abstractions composables pour le machine learning dynamique** L'approche RELAX de Lai *et al.* (2023) propose des abstractions modulaires facilitant l'intégration de modèles de machine learning dynamiques sur des plateformes embarquées. Grâce à ces abstractions composables, il devient possible de concevoir des pipelines de machine learning capables de s'adapter en temps réel aux variations des environnements d'exécution et aux contraintes spécifiques, ce qui est particulièrement pertinent pour des applications nécessitant une flexibilité continue, comme dans le secteur médical ou domotique.

Ces outils et approches (AutoTVM, MATCH, MicroNets et RELAX) montrent que l'avenir du TinyML repose sur une synergie étroite entre l'optimisation du compilateur et la conception de modèles adaptés aux contraintes matérielles. En intégrant ces techniques dans des workflows de développement automatisés, il devient possible de générer des modèles optimisés pour une vaste gamme de dispositifs, garantissant ainsi des performances accrues tout en respectant les contraintes sévères imposées par les systèmes embarqués.

Ces avancées ouvrent la voie à des systèmes intelligents et autonomes capables d'exécuter des modèles complexes dans des environnements contraints, ce qui est déterminant pour des applications modernes et critiques, notamment dans le domaine de la santé.

CHAPITRE 5

ENJEUX ÉTHIQUES ET ACCEPTABILITÉ SOCIALE

L'essor des technologies d'apprentissage automatique transforme radicalement notre manière de traiter l'information et d'innover dans divers secteurs. Si ces avancées offrent des opportunités considérables, elles soulèvent également d'importantes questions éthiques et sociétales. La protection des données personnelles, la sécurité des systèmes et la vérification de la provenance des données utilisées pour entraîner les modèles sont des défis techniques qui nécessitent une vigilance constante.

Par ailleurs, des enjeux liés à la responsabilité dans les décisions automatisées viennent complexifier l'utilisation de ces technologies. Enfin, la manière dont ces outils sont perçus par le public, leur impact sur les pratiques sociales et leur acceptabilité générale restent des questions centrales pour assurer une intégration harmonieuse et responsable dans la société.

5.1 Confidentialité, sécurité et provenance des données

Les algorithmes d'apprentissage automatique ont besoin de grandes quantités de données pour fonctionner. Ces données sont souvent sensibles, comme des informations sur la santé, des images de personnes ou des comportements sur internet. Si ces technologies apportent des bénéfices importants, elles présentent aussi des risques pour la vie privée. Les questions de confidentialité, de sécurité et d'origine des données sont donc essentielles pour s'assurer que ces outils soient utilisés de manière responsable.

5.1.1 Protection et sécurité des données personnelles

Les données personnelles correspondent à toute information permettant d'identifier une personne. Cela peut être sous la forme d'un nom, une adresse, des antécédents médicaux ou même ses habitudes sur internet. Lorsqu'elles sont utilisées pour entraîner des algorithmes, elles doivent être protégées, car leur fuite peut avoir de graves conséquences.

Des lois ont été mises en place dans plusieurs pays pour encadrer l'utilisation de ces données sensibles. En Europe, le règlement général sur la protection des données (RGPD) impose l'accord des personnes avant la collecte de leurs informations. Il permet aussi d'avoir le droit de savoir comment les données sont utilisées et de demander leur suppression (Bufalieri, Morgia, Mei & Stefa (2020)).

En plus des lois, des outils techniques permettent de limiter les risques en anonymisant ou pseudonymisant les informations, ce qui empêche de retrouver l'identité de la personne à l'origine des données. Des méthodes comme l'anonymisation des bases de données tabulaires visent à garantir la sécurité des données tout en conservant leur utilité pour l'entraînement des algorithmes (Nguyen & Castelluccia (2020)). De plus, des techniques de chiffrement sont aujourd'hui mises en place pour rendre les données illisibles en cas de fuite.

Néanmoins, l'une des méthodes permettant la protection la plus efficace est l'apprentissage fédéré. Cela signifie que les données ne sont jamais transmises. Elles servent à entraîner un algorithme directement sur un appareil local, et c'est uniquement le modèle mis à jour qui est envoyé vers un serveur. Cela garantit que les données personnelles restent sur l'appareil de l'utilisateur, réduisant ainsi considérablement les risques de fuite (Truong, Sun, Wang, Guitton & Guo (2021)).

5.1.2 Provenance des données et risque de vol

Il est crucial de savoir d'où proviennent les données utilisées pour entraîner les algorithmes, car celles-ci peuvent être de mauvaise qualité, fausses, incomplètes ou collectées de manière illégale.

Dans certains cas, des entreprises ont utilisé des photos de personnes trouvées sur Internet sans leur consentement pour entraîner des algorithmes de reconnaissance faciale. Par exemple, la société Clearview AI a été accusée d'avoir collecté des milliards d'images sur des plateformes comme Facebook et LinkedIn sans l'accord des utilisateurs (Longpre *et al.* (2023)). De même, des algorithmes génératifs comme GPT ou MidJourney se sont appuyés sur d'immenses quantités

de données disponibles en ligne, souvent sans que les créateurs de contenu aient donné leur autorisation (Longpre *et al.* (2024)). Ces pratiques sont controversées car elles portent atteinte à la vie privée et aux droits d'auteur, et elles peuvent aussi servir à des fins malveillantes.

Pour éviter des problèmes juridiques, éthiques ou de qualité des données, il est essentiel de vérifier la provenance des données. Cela implique de garder une trace de leur origine et de s'assurer qu'elles ont été obtenues de manière légale. La traçabilité des données est l'un des éléments fondamentaux concernant l'éthique et la qualité des bases de données (Whitney & Norman (2024)).

Enfin, l'utilisation de données synthétiques, générées artificiellement par des algorithmes, se développe pour limiter la manipulation de données réelles sensibles. Ces données permettent également d'obtenir en grande quantité des ensembles variés et adaptés à l'entraînement de modèles auto-entraînés (Hawkins & Mittelstadt (2023)).

Bien que cette solution soit prometteuse, elle reste en phase de développement et ne remplace pas encore complètement les données authentiques.

5.2 Éthique et transparence dans l'utilisation des algorithmes

L'utilisation croissante d'algorithmes d'apprentissage automatique dans des domaines sensibles comme la santé, la finance ou encore la gestion des ressources humaines a mis en lumière des enjeux éthiques majeurs. Ces technologies, bien qu'efficaces, ne sont pas exemptes de dérives et soulèvent des questions sur l'équité, la responsabilité et l'impact environnemental de leur déploiement.

5.2.1 Biais algorithmiques et discrimination

Les biais algorithmiques constituent l'un des principaux risques éthiques associés aux systèmes d'apprentissage automatique. Ces biais apparaissent lorsque les données utilisées pour entraîner les modèles reflètent des inégalités sociales, des stéréotypes ou des déséquilibres structurels

(Wachter, Mittelstadt & Russell (2021)). Par exemple, dans le domaine médical, un algorithme peut se révéler moins précis pour certaines populations en raison d'un sous-échantillonnage de celles-ci dans les données initiales (Chen, Johansson & Sontag (2018b)). De même, dans les systèmes de recrutement automatisés, des biais sexistes ou raciaux peuvent conduire à discriminer certains candidats.

Ces biais peuvent résulter d'une collecte de données non représentative, de l'existence d'erreurs et d'approximations dans les bases de données, ou encore d'hypothèses directement intégrées durant la conception des algorithmes (Ferrer, Nuenen, Such, Cote & Criado (2021)).

Les conséquences de ces erreurs peuvent être lourdes, pouvant mener à des décisions injustes, voire discriminatoires. Des cas d'IA adoptant des comportements racistes ou sexistes ont déjà été observés, souvent en raison d'un manque de supervision et de contrôle rigoureux autour de leur développement (Hong & Williams (2019); Wachter *et al.* (2021)).

Il est donc essentiel de mettre en place des mécanismes d'évaluation continue pour détecter et corriger ces biais. Des approches comme l'audit algorithmique ou l'intégration de métriques d'équité sont de plus en plus utilisées pour limiter ces dérives.

5.2.2 Responsabilité

La délégation de certaines décisions à des systèmes automatisés soulève également la question de la responsabilité. Lorsqu'un algorithme produit une erreur ayant des répercussions graves (comme un mauvais diagnostic médical ou le refus d'un crédit injustifié), il est souvent difficile d'identifier précisément les responsables (Mittelstadt, Allo, Taddeo, Wachter & Floridi (2016)). S'agit-il du concepteur de l'algorithme, de l'organisation qui l'a déployé ou de l'utilisateur final ?

Cette dilution de la responsabilité est d'autant plus problématique que certains modèles d'apprentissage profond, en particulier les réseaux neuronaux, fonctionnent comme des "boîtes noires". Leurs décisions sont parfois difficiles à interpréter, ce qui complique la compréhension

des causes d'une erreur. Cette opacité pose un défi majeur pour garantir la redevabilité et la confiance des utilisateurs.

Pour répondre à ces enjeux, des efforts sont menés afin de rendre les algorithmes plus explicables et compréhensibles. Des techniques d'explicabilité, comme LIME (Local Interpretable Model-agnostic Explanations) ou SHAP (SHapley Additive Explanations), permettent de mieux comprendre les facteurs ayant influencé une prédiction donnée (Ribeiro, Singh & Guestrin (2016); Lundberg & Lee (2017)). Ces approches visent à renforcer la transparence et à clarifier la chaîne de responsabilité.

5.2.3 Impact environnemental et durabilité

L'essor des modèles d'apprentissage automatique, en particulier ceux de grande taille comme les réseaux neuronaux profonds et les modèles de traitement du langage naturel (comme GPT, Claude, etc), a mis en lumière leur impact environnemental préoccupant. La consommation énergétique liée à l'entraînement et au déploiement de ces modèles est devenue une source majeure d'inquiétude dans la communauté scientifique, notamment en raison de l'empreinte carbone associée aux centres de données et aux infrastructures de calcul.

L'entraînement de ces modèles est particulièrement énergivore. Patterson *et al.* (2021) estiment que l'entraînement de GPT-3, contenant 175 milliards de paramètres, a consommé environ 1 287 MWh d'électricité, générant 502 tonnes de CO_2e, soit l'équivalent de plusieurs centaines de vols transatlantiques. Ce coût énergétique est accentué par l'utilisation d'accélérateurs matériels (GPU, TPU), qui bien qu'optimisés pour les calculs massivement parallèles, nécessitent des infrastructures spécifiques et gourmandes en ressources.

De plus, les émissions de carbone liées à l'IA dépendent fortement de l'emplacement géographique des centres de données et du mix énergétique local. Henderson *et al.* (2022) soulignent que la même tâche d'apprentissage automatique effectuée dans une région alimentée majoritairement par des énergies fossiles peut avoir une empreinte carbone jusqu'à 30 fois plus élevée que si elle était exécutée dans une région utilisant des énergies renouvelables. Ce constat

met en évidence l'importance de bien choisir les lieux d'entraînement des modèles pour limiter leur impact environnemental.

Pour répondre à ces préoccupations, Schwartz, Dodge, Smith & Etzioni (2020) plaident pour une approche dite "Green AI", qui valorise une IA sobre en ressources, privilégiant l'efficacité énergétique et la réduction des coûts environnementaux plutôt que l'optimisation extrême des performances. Cette approche encourage la publication systématique de métriques sur la consommation énergétique des modèles afin de sensibiliser les chercheurs et les industriels. Il considérer l'efficacité environnementale comme une métrique aussi importante que la précision des modèles.

Enfin, certaines entreprises et laboratoires adoptent désormais des pratiques plus responsables, telles que l'utilisation de serveurs alimentés par des énergies renouvelables, la mutualisation des ressources de calcul ou encore la recherche d'algorithmes moins gourmands en données et en temps d'entraînement.

Cependant, comme le souligne Schwartz *et al.* (2020), l'empreinte carbone de l'IA soulève également des questions d'équité et d'inégalité d'accès aux ressources computationnelles. Les entreprise disposant de moyens financiers importants peuvent se permettre d'entraîner des modèles de plus en plus complexes, tandis que les équipes de recherche académique ou les petites structures peinent à rivaliser. Cette concentration des ressources accentue le déséquilibre entre les grandes entreprises technologiques et le reste des acteurs de la recherche.

Ainsi, si les algorithmes d'apprentissage automatique offrent des perspectives prometteuses, leur coût environnemental impose une réflexion sur leur durabilité. L'avenir passe sans doute par une IA plus sobre, intégrant dès sa conception des critères d'efficacité énergétique et d'empreinte carbone réduite.

L'intégration systématique d'indicateurs de consommation énergétique dans les publications, ainsi que la transparence sur les coûts environnementaux des modèles, constituent des étapes clés pour encourager une transition vers une intelligence artificielle durable.

5.3 Acceptabilité sociale et impacts sociétaux

Les avancées rapides dans le domaine de l'apprentissage automatique transforment progressivement de nombreux secteurs, allant de la santé à l'agriculture, en passant par l'industrie. Cependant, l'acceptation de ces technologies par la société ne suit pas toujours le même rythme. L'adoption de solutions fondées sur l'IA dépend non seulement de leur efficacité technique, mais aussi de la manière dont elles sont perçues par les citoyens et intégrées dans leurs pratiques quotidiennes. La dimension humaine est donc déterminante pour assurer une transition réussie vers des systèmes d'IA socialement acceptables.

5.3.1 Perception et confiance du public

La perception du public vis-à-vis les technologies d'apprentissage automatique change souvent entre fascination et méfiance. D'une part, les performances impressionnantes des algorithmes dans des domaines comme la reconnaissance d'images, les IA génératives ou l'aide au diagnostic médical suscitent l'enthousiasme. D'autre part, les erreurs spectaculaires, les biais discriminatoires et les scandales liés à l'exploitation des données personnelles renforcent la méfiance de la population (Jobin, Ienca & Vayena (2019)).

La confiance accordée à ces technologies repose principalement sur trois facteurs : la transparence, la sécurité des données et la gestion des erreurs algorithmiques. Comme l'a montré l'étude de Scantamburlo *et al.* (2023), cette confiance est renforcée lorsque les décisions assistées par l'IA restent supervisées par un humain, que des garanties claires sur la sécurité des données sont assurées, et que les utilisateurs disposent d'informations sur le fonctionnement général des algorithmes.

Dans le domaine médical, par exemple, l'acceptabilité est nettement meilleure lorsque l'intelligence artificielle est perçue comme un outil venant soutenir le professionnel de santé, et non comme un système autonome prenant des décisions critiques à sa place (Lysaght, Lim, Xafis & Ngiam (2019)). Ainsi, la transparence, la protection des données et la présence

d'un contrôle humain apparaissent comme des leviers fondamentaux pour favoriser une adoption sereine des technologies d'apprentissage automatique au sein de la société.

Un autre facteur, souvent sous-estimé, réside dans le choix des termes utilisés pour désigner ces technologies. Comme le souligne Langer, Hunsicker, Feldkamp, König & Grgić-Hlača (2022), il est établi que le terme *algorithme* est généralement mieux accepté par le public que *intelligence artificielle* ou *IA*. Le mot algorithme évoque une notion plus familière, souvent perçue comme une simple formule mathématique fiable et neutre. À l'inverse, IA est fréquemment associé à des concepts futuristes ou à des scénarios dystopiques véhiculés par la culture populaire, ce qui peut créer des craintes.

Cette distinction est d'autant plus importante que, dans les faits, l'intelligence artificielle est présente dans notre quotidien depuis plusieurs décennies. Les algorithmes d'apprentissage automatique sont utilisés depuis longtemps dans des services tels que les recommandations sur les plateformes de streaming, la surveillance médicale ou encore le calcul d'itinéraires GPS. Pourtant, l'apparition récente et massive du terme "IA" dans l'espace médiatique a donné l'impression d'une rupture technologique radicale, alors qu'il s'agit souvent d'une évolution progressive de techniques déjà bien intégrées.

Ainsi, il est essentiel de comprendre que l'intelligence artificielle est perçue de manière plus favorable lorsqu'elle est présentée comme une continuité naturelle des solutions algorithmiques existantes. À l'inverse, les systèmes qui introduisent des modes d'interaction radicalement nouveaux avec le public peuvent susciter davantage de réticences.

5.3.2 Impact sur les pratiques sociales et communication

Au cours des dernières années, l'essor des algorithmes d'apprentissage automatique a progressivement transformé les pratiques sociales et les modes de communication. Ces technologies ont modifié la nature du travail dans de nombreux secteurs, automatisant certaines tâches et alimentant les inquiétudes quant à une potentielle perte d'emplois ou une déqualification des travailleurs. Dans le domaine médical, par exemple, l'introduction de l'IA a redéfini la relation entre soignants

et patients. Désormais, ces outils permettent d'améliorer la précision des diagnostics tout en faisant émerger une forme de dépendance aux algorithmes automatiques (Jiang *et al.* (2021)).

Bien que ces évolutions suscitent parfois des craintes légitimes, ils ouvrent également la voie à de nouvelles opportunités. L'intelligence artificielle, lorsqu'elle est développée et utilisée de manière responsable, peut devenir un puissant levier de progrès social. Elle est en mesure de libérer du temps pour des tâches à plus forte valeur humaine, d'améliorer l'accès aux soins médicaux, et d'accompagner les professionnels dans leur travail quotidien.

CHAPITRE 6

APPLICATION

6.1 Optimisation de masque KN95

Au cours de l'épidémie de Covid-19, l'importance du suivi de la santé respiratoire des patients a été largement démontrée. Les masques KN95 ont été largement utilisés, offrant une protection efficace contre les contaminants tels que les bactéries et les virus aéroportés. Cependant, bien qu'efficaces pour la filtration, ces masques assurent uniquement une protection passive et ne fournissent aucune information sur l'état respiratoire de l'utilisateur. Les masques KN95 sont conçus pour offrir une capacité de filtration de 95% des particules de taille supérieure à 3 microns.

Dans le cadre de ce projet de recherche, pour aller au-delà de cette simple fonction de protection, un système de capteurs a été intégré aux masques afin de caractériser leur efficacité. Cette matrice de capteurs permet de mesurer le taux d'humidité piégé dans les fibres du masque, ce qui offre deux avantages : évaluer la capacité filtrante du masque au fil du temps, et suivre les variations d'humidité dues à la respiration, permettant ainsi un suivi respiratoire en temps réel.

Pour analyser ces données, un modèle non supervisé a été développé afin d'identifier les patterns respiratoires spécifiques à chaque type de respiration effectuée durant les phases de test. Ce modèle permet non seulement de réaliser un premier filtrage des données, mais aussi de détecter d'éventuelles anomalies respiratoires.

En complément, un second modèle supervisé a été conçu pour classifier les différents schémas respiratoires identifiés. Ce modèle a été optimisé pour être aussi compact que possible, afin de pouvoir être intégré, si nécessaire, dans un dispositif portable embarqué, garantissant ainsi une surveillance continue et discrète de la santé respiratoire.

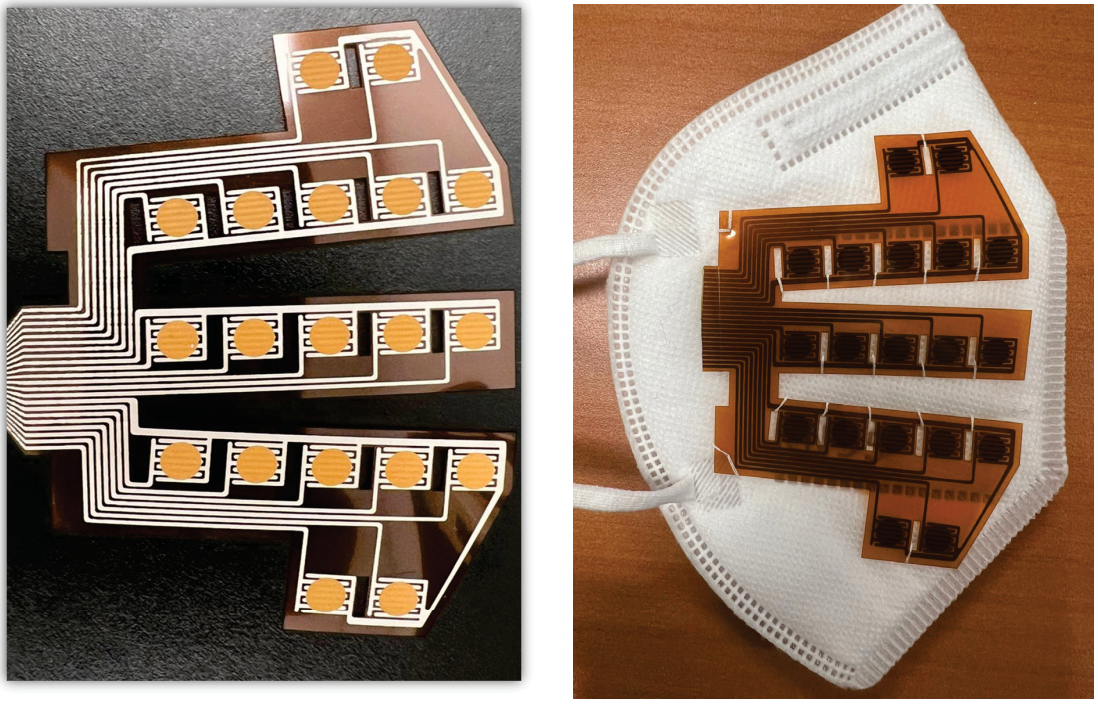
CHAPITRE 7

MÉTHODOLOGIE

7.1 Matrice de capteurs

La matrice de capteurs pour notre masque est réalisée par sérigraphie. Cette méthode permet de créer des circuits et des capteurs flexibles qui épousent parfaitement la forme d'un masque KN95. Les capteurs utilisés sont des capteurs d'humidité à base de $BiFeO_3$, dont la fabrication est décrite dans le travail de Fourmont *et al.* (2024). Ces capteurs présentent une réponse inverse, ce qui signifie que leur résistance diminue à mesure que l'humidité augmente.

La matrice de capteurs est constituée de 19 capteurs par face, soit un total de 38 capteurs. Chaque matrice de 19 capteurs a été arrangée de manière à suivre la forme légèrement triangulaire du masque, comme illustré sur la figure 7.1b.



a) Matrice de capteur b) Matrice cousu sur un masque

Figure 7.1 Matrice de capteur appliqué à un masque KN95

Chaque capteur est composé de deux éléments : une électrode et un élément actif. L'électrode est sérigraphiée à partir de l'encre d'argent EDAG 725A de Henkel sur une feuille de polyimide. L'encre est d'abord séchée à l'air libre, puis cuite dans un four à 300°C pendant une heure. Le circuit obtenu comprend les électrodes ainsi que les pistes de connexion nécessaires à la connectique.

Sur cette première couche, on dépose ensuite l'encre à base de $BiFeO_3$. Cette couche est également séchée à l'air, puis recuite au four à 300°C pendant 10 minutes.

Le circuit final est ensuite découpé au laser à l'aide d'un LPKF Protolaser U3. Cette étape permet de retirer l'excédent de matière du substrat et de créer des ouvertures supplémentaires pour s'assurer que la matrice de capteurs n'obstrue pas le passage de l'air, garantissant ainsi le confort respiratoire de l'utilisateur.

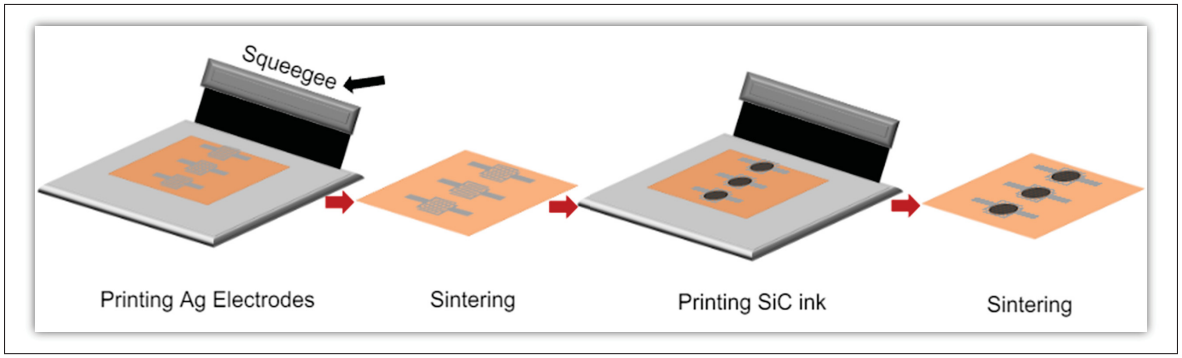


Figure 7.2 Fabrication de capteur par sérigraphie. (1) sérigraphie du circuit. (2) séchage et frittage de l'encre. (3) sérigraphie du matériau actif. (4) séchage et frittage (ou recuit) du matériau actif

Une carte d'interfaçage a été conçue pour pouvoir relier les matrices de capteurs à l'appareil de mesure. Cette carte utilise un connecteur enfichable permettant de faciliter au maximum la connexion entre les matrices et l'appareil de mesure. Ce type de connecteur utilise des lamelles de métal qui viennent pincer le circuit pour faire la connexion. Cependant, la pression lors de l'insertion du circuit peut venir légèrement endommager les traces d'argent de notre circuit pouvant, en cas de nombreuses utilisations, réduire la qualité de la connexion voire même la perte totale de celle-ci. L'épaisseur de notre substrat au niveau du connecteur peut donc être

ajustée à l'aide de film de polyimide pour ajuster son épaisseur et donc la pression excercée sur les traces d'argent.

L'appareil de mesure est un système d'acquisition et d'enregistrement de données Keithley DAQ6510, avec le module enfichable 7708 permettant d'avoir accès à 40 canaux multiplexés.

7.2 Protocole expérimental

Les mesures ont été effectuées sur une période de trois jours afin de garantir la répétitivité des résultats et d'observer la possible dégradation des capteurs ainsi que celle des masques. De plus, la température ainsi que hygrométrie de la pièce ont été contrôlées pour effectuer les mesures dans les mêmes conditions environnementales.

Chaque participant a dû réaliser cinq cycles de respiration, alternant entre des phases de respiration normale et de respiration profonde.

Pour s'assurer de la stabilité des capteurs, une période de deux minutes à l'air libre a été observée avant chaque session, permettant aux capteurs de fournir une valeur de référence à température ambiante. Les cycles de respiration se déroulent de la manière suivante : 5 minutes de respiration profonde, suivie de 5 minutes de respiration normale.

7.3 Prétraitement des données

Le traitement et l'analyse des données ont été réalisés dans un environnement Python. Les données obtenues à partir des capteurs d'humidité ainsi que celles provenant du masque KN95 présentent un comportement suivant une courbe logarithmique inverse. Il est donc nécessaire d'aplatir cette courbe pour un meilleur traitement des données.

Pour ce faire, la courbe est d'abord segmentée en portions de 64 points de données, représentant approximativement 30 secondes. Une régression linéaire est ensuite appliquée à chaque segment à l'aide de la librairie scikit-learn afin d'extraire la tendance de chaque portion de la courbe. Cette tendance est ensuite soustraite aux données d'origine, ce qui permet de centrer chaque

segment autour de zéro. En appliquant ce processus à l'ensemble des segments, la courbe des données est redressée. Le signal résultant est ainsi centré sur zéro tout au long de la prise de mesure.

L'approche par segmentation a été choisie car la réponse des capteurs à l'humidité varie en fonction des conditions initiales des tests, notamment en fonction de l'âge des capteurs et des caractéristiques du patient.

Les valeurs aberrantes sont ensuite supprimées à l'aide de la méthode de l'écart interquartile (IQR). Tous les points situés à plus de 1,5 fois l'IQR en dessous du premier quartile ou au-dessus du troisième quartile sont retirés. Les données sont ensuite normalisées selon la méthode min-max afin de les ramener sur une même échelle, rendant ainsi les caractéristiques comparables (Klambauer *et al.* (2017)).

Les caractéristiques des données de respiration sont ensuite extraites à l'aide d'une fenêtre glissante de 10 secondes, appliquée sur l'ensemble des données avec un pas d'une unité. Pour chaque fenêtre, les caractéristiques suivantes sont calculées : la moyenne, la variance, l'amplitude, le skewness, ainsi que la puissance spectrale pour les fréquences inférieure à 0.5Hz.

Le skewness est une mesure de l'asymétrie d'un signal périodique, calculée selon la formule suivante :

$$\mathrm{Sk} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \frac{(x_i - \overline{x})^3}{\sigma^3} \tag{7.1}$$

où n représente la taille de l'échantillon, x_i sont les valeurs de l'échantillon, $\overline{x}$ est la moyenne de l'échantillon, σ est l'écart-type.

La puissance spectrale est quant à elle calculée à partir de la transformée de Fourier rapide (FFT). Ensuite, un algorithme de clustering appelé HDBSCAN est utilisé. Cet algorithme, basé sur la densité, permet de déterminer un nombre optimal de clusters sans avoir besoin de le spécifier à l'avance. Cette approche est privilégiée car nos données peuvent être bruitées, en particulier lors

de la phase de stabilisation des courbes de respiration. HDBSCAN permet d'obtenir des clusters plus petits mais plus fiables.

Le nombre de clusters attendu est de quatre : respiration normale, respiration profonde, absence de respiration ou respiration irrégulière, et le bruit.

Les clusters sont ensuite visualisés dans un espace 2D en appliquant la méthode t-SNE pour validation. Les données bruitées, labellisées -1 par HDBSCAN, sont retirées pour ne conserver que les clusters denses. Les données restantes sont étiquetées comme suit : "Normal", "Irregular", et "Deep".

L'ensemble de ces manipulations vise à mettre en place une détection automatique des types de schémas respiratoires effectués durant les tests, afin de fournir les données nécessaires à la construction d'un modèle de classification supervisé. Cette approche permet à la fois de nettoyer les données en éliminant une partie du bruit et de détecter de potentielles anomalies qui pourraient influencer le modèle de classification lors de l'entraînement.

7.4 Deployement des modèles TinyML

Deux modèles ont été déployés sur une Arduino Nano 33 BLE, équipée du microcontrôleur nRF52840 de Nordic Semiconductor. Le premier vise à faire une classification des schémas respiratoires à partir des classes obtenues par le modèle de clustering. Le deuxième est un modèle de classification utilisant la même architecture que le premier mais avec des dimensions bien plus grandes. Ce plus gros modèle permettra de pouvoir montrer l'impact des optimisations lors du déploiement.

Le premier modèle est un réseau de neurones composé d'un vecteur d'entrée de dimension 1x5, suivi de deux couches denses de 16 neurones chacune. La couche d'entrée transmet le vecteur à la première couche cachée. Chaque neurone de cette couche applique un ensemble de poids et un biais au vecteur d'entrée, puis le résultat est passé à une fonction d'activation ReLU. ReLU est choisie car elle ne requiert qu'une opération de comparaison et un maximum, réduisant

ainsi la charge de calcul par rapport à des fonctions plus complexes comme tanh ou sigmoid, ce qui est essentiel pour des applications embarquées à ressources limitées. Les résultats obtenus constituent la sortie de la première couche d'activation. La deuxième couche cachée répète le même processus, en prenant comme entrée la sortie de la première couche. Enfin, la sortie de la seconde couche est transmise à une couche softmax.

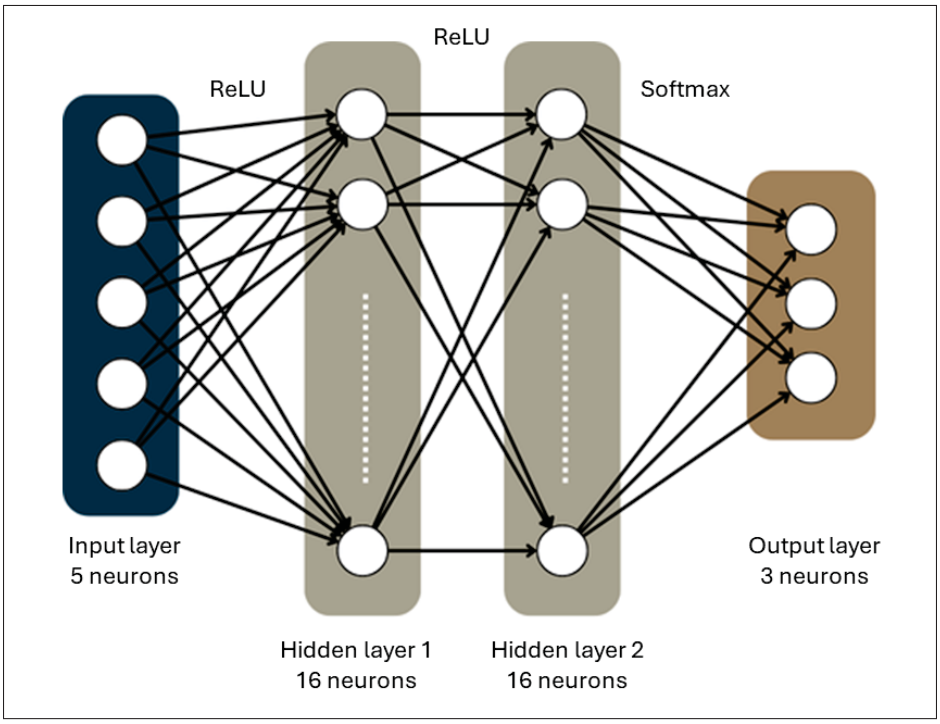


Figure 7.3 Réseaux de neurones du premier modèle

La fonction softmax ($\sigma(z)_j$) agit comme une fonction de normalisation, transformant le vecteur d'entrée **z**, composé de K dimensions avec des valeurs réelles, en un vecteur de valeurs réelles compris dans l'intervalle $[0, 1]$, dont la somme est égale à 1. Plus précisément, cela peut être interprété comme une distribution de probabilité sur les 3 classes de sortie.

Le deuxième modèle est également un réseau de neurones, mais avec un vecteur d'entrée de dimension 1x851. Il est suivi de deux couches denses comportant chacune 20 neurones. Comme dans le premier modèle, les couches cachées utilisent la fonction d'activation ReLU, tandis que

la sortie de la seconde couche est passée à une fonction softmax appliquée sur les 250 classes de sortie.

Les modèles de classification sont ensuite convertis via TensorFlow Lite, où ils sont quantifiés de float32 à int8, tout en conservant les entrées et sorties en float32. Ces modèles sont ensuite implémentés sous forme de fichiers d'en-tête C et déployés à l'aide de TVM sur TinyML.

TVM alloue un bloc mémoire pour stocker les paramètres du modèle, ainsi que les tenseurs d'entrée, de sortie et intermédiaires nécessaires à l'exécution. Un interpréteur TVM gère le flux de travail en traitant les données en entrée, en exécutant le modèle, puis en récupérant les prédictions. Le système fonctionne de manière séquentielle : Les données entrantes sont d'abord stockées dans une mémoire tampon interne, quantifiées une fois leur réception complète, elles sont ensuite traitées par le modèle. Enfin, le tampon d'entrée est vidé pour préparer la prochaine mesure.

TVM Relay (Chen *et al.* (2018c)) est utilisé pour optimiser les opérations du réseau de neurones. TVM applique une correspondance de motifs (pattern matching) afin d'identifier les opérations courantes pouvant être fusionnées. Il vérifie également que la fusion des couches n'impacte pas la précision des calculs et que ces optimisations sont spécifiquement adaptées à la cible nRF52840. Les différentes étapes de cette procédure sont décrites dans l'algorithme 27.

Pour améliorer davantage la gestion de la mémoire SRAM, les modèles sont partitionnés afin de transférer de façon séquentielle les poids de la mémoire Flash vers la SRAM. Cette approche permet d'optimiser l'interaction entre la SRAM et la mémoire Flash sur le microcontrôleur, en minimisant l'utilisation de la SRAM tout en garantissant des performances d'inférence efficaces. Ce processus est particulièrement utile pour les systèmes embarqués avec des contraintes de mémoire strictes, comme ceux utilisant le microcontrôleur nRF52840.

L'algorithme décrivant cette procédure est présenté dans l'algorithme 19.

Algorithme 7.1 Déploiement du modèle

Algorithme : Déploiement des modèles
Input : $x(n)$: 1D spectrum of length N
Output : `Deployed_Model` : Optimized model for `nrf52840`

Quantization :;
$Q \leftarrow \emptyset$; /* Quantization parameters */
foreach *layer L in Model* **do**
  $\mu \leftarrow \text{mean}(L_{\text{weights}})$; /* Calculate mean */
  $\sigma \leftarrow \text{std}(L_{\text{weights}})$; /* Calculate standard deviation */
  $s \leftarrow \frac{\max(|L_{\text{weights}}|)}{2^7}$; /* Find quantization scale for INT8 */
  $L_q \leftarrow \text{round}(L_{\text{weights}}/s)$; /* Quantize weights */
  $Q \leftarrow Q \cup \{s\}$; /* Store scale factors */
end foreach
TVM Relay Optimization :
  $G \leftarrow \text{BuildGraph}(\text{Model}_q)$; /* Build computational graph */
$P \leftarrow \text{IdentifyPatterns}(G)$; /* Find fusion patterns */
foreach *p in P* **do**
  if *DependencyCheck*(p) **then**
    $G \leftarrow \text{FuseOperators}(G, p)$; /* Fuse compatible operators */
  end if
end foreach
Memory Allocation :
  $M \leftarrow \text{CalculateMemoryRequirements}(G)$;
if $M \leq$ *nrf52840_memory* **then**
  AllocateMemory(G);
end if
else
  return Error : "Memory exceeded";
end if
$C \leftarrow \text{GenerateCode}(G)$; /* Generate C code */
Binary $\leftarrow \text{Compile}(C, \text{"nrf52840"})$; /* Compile for target */
return `Binary`;

Algorithme 7.2 Gestion de la mémoire et exécution du modèle

Algorithme : Gestion de la mémoire et exécution du modèle
Input : Graph G representing the model dependencies.
Output : Binary compiled for the target (`nrf52840`).

$P \leftarrow$ PartitionModel(G) ; /* Partition model by layers */
flash_addr $\leftarrow$ StoreWeights(P) ; /* Store all weights in Flash */
foreach *partition* $p_i \in P$ **do**
    Memory Transfer :
     required_sram $\leftarrow$ CalculatePartitionMemory(p_i);
    if *required_sram* > *available_sram* **then**
        return Error("Partition too large for SRAM");
    end if
    Load Weights :
     sram_addr $\leftarrow$ AllocateSRAM(required_sram);
    LoadFromFlash(flash_addr[p_i], sram_addr);
    Execute Partition :
     intermediate_output $\leftarrow$ ExecutePartition(p_i, sram_addr);
    Clean Up :
     FreeSRAM(sram_addr) ; /* Clear SRAM for next partition */
    if $p_i \neq$ *last_partition* **then**
        StoreIntermediate(intermediate_output) ; /* Store next partition */
    end if
end foreach
$C \leftarrow$ GeneratePartitionedCode(P) ; /* Generate memory-aware C code */
Binary $\leftarrow$ Compile(C, "nrf52840") ; /* Compile for target */
return Binary;

CHAPITRE 8

ANALYSE DES RÉSULTATS ET DISCUSSION

Dans cette partie sont présentés les résultats obtenus sur le projet du masque intelligent, ainsi que les méthodes d'optimisation et de réduction des modèles pour les rendre compatibles avec l'Arduino Nano 33 BLE.

8.1 Masque

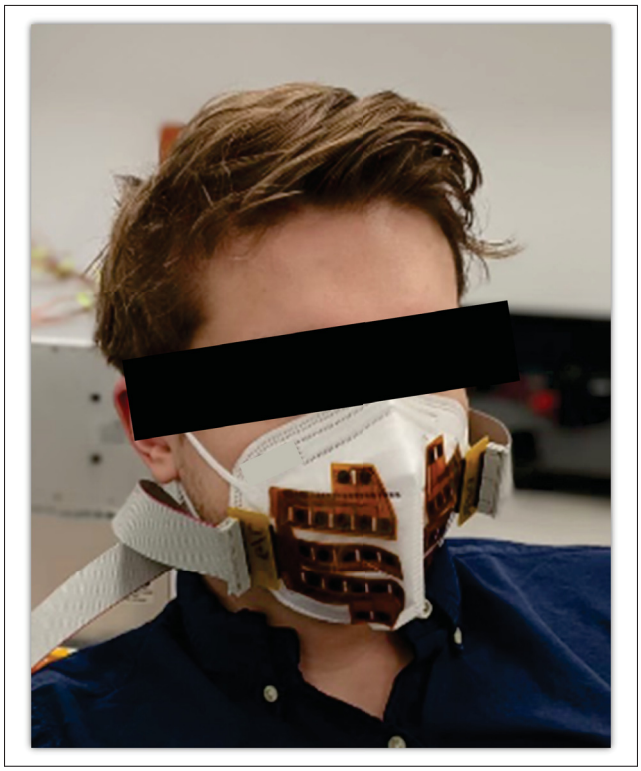


Figure 8.1 Test de respiration

Les capteurs d'humidité embarqués sur le masque permettent de mesurer les variations d'humidité générées par la respiration. Ces variations évoluent au fil du temps en raison de l'accumulation de micro-gouttelettes d'eau piégées dans les fibres du masque. À mesure que la quantité d'eau emprisonnée augmente, l'humidité moyenne à l'intérieur du masque ainsi qu'à sa périphérie extérieure s'élève également.

Cette accumulation a pour conséquence de produire une réponse logarithmique inverse dans le signal capté. Cette réponse est composée de deux éléments distincts : l'aspect inversé, qui est intrinsèque au type de capteur utilisé, et l'aspect logarithmique, qui dépend de l'augmentation progressive de l'humidité piégée dans le masque. Cette réponse peut être observée sur la figure 8.2.

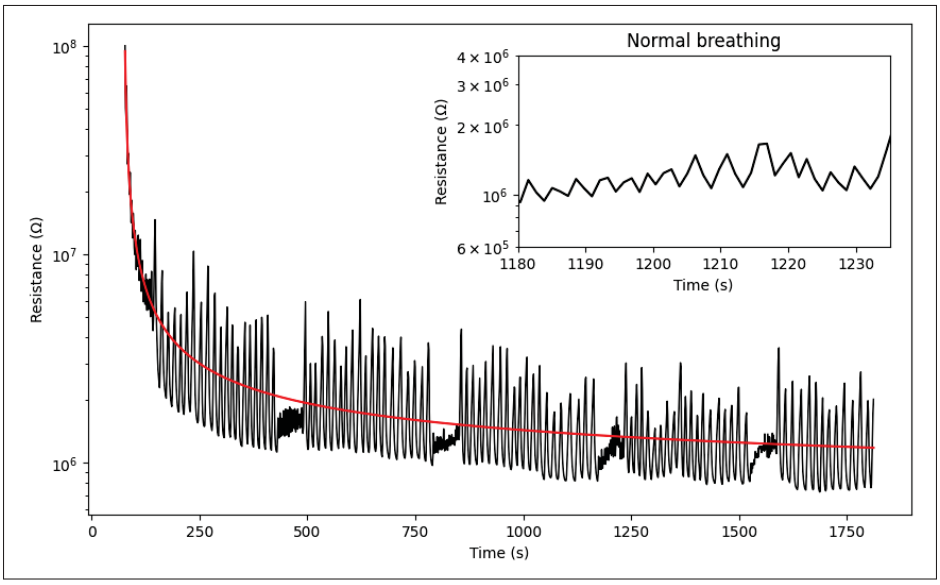


Figure 8.2 Courbe de la réponse des capteurs d'humidité à un test de respiration

La matrice de capteurs permet d'accéder à une variété de positions où la réponse est plus ou moins prononcée, en fonction de la localisation des capteurs : qu'ils soient placés en face de la bouche ou sur les côtés du masque.

Cependant, en raison de l'aspect logarithmique du signal, les sinusoïdes respiratoires sont fortement déformées, rendant leur interprétation impossible tant que le signal n'est pas stabilisé. Cette stabilisation intervient lorsque l'équilibre est atteint et que le taux d'humidité dans le masque se stabilise. Le temps nécessaire pour atteindre cette stabilité dépend fortement de l'individu et du type de respiration effectuée.

Comme montré sur la figure 8.2, la différenciation entre le bruit et le signal respiratoire intervient après une minute. En revanche, sur la figure 8.3, cette différenciation apparaît après environ

deux minutes. Lors de ces tests, il a été observé que le premier exercice de respiration profonde apporte une grande quantité d'humidité, ce qui permet d'obtenir une stabilisation plus rapide du signal.

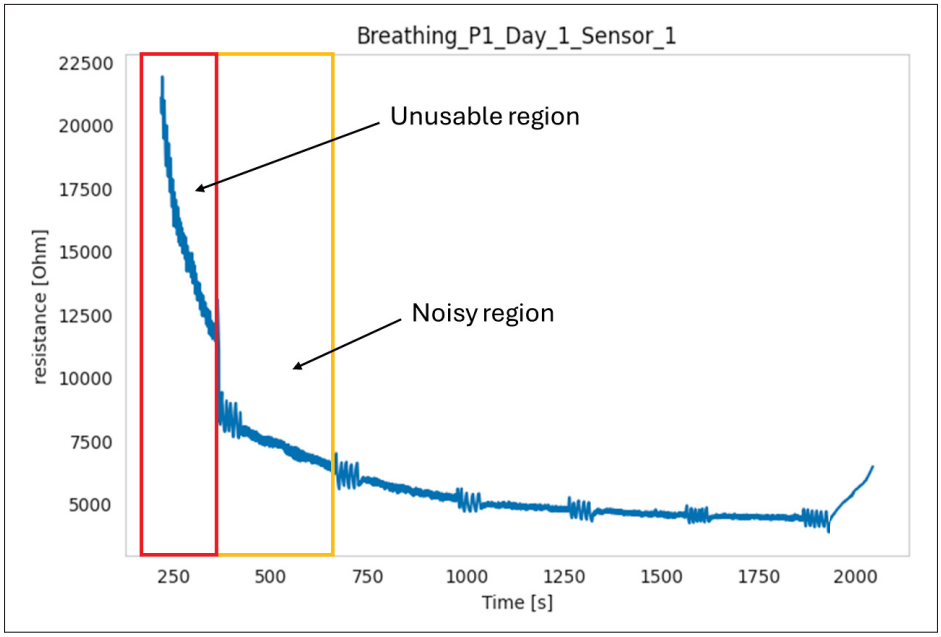


Figure 8.3 Réponse bruité lors de la stabilisation du signal respiratoire

Le signal, étant trop bruité sur sa partie descendante, est partiellement retiré afin d'éviter de détériorer la qualité des données. En revanche, la région bruitée est conservée, car le bruit sera en partie filtré lors de la phase de traitement. Il est à noter que la partie linéaire en fin de signal est également conservée, car elle permet de détecter une absence de respiration. Les caractéristiques extraites du signal conservé sont les suivantes : la moyenne, la variance, l'amplitude, le skewness et la puissance des basses fréquences.

Ces caractéristiques sont utilisées pour appliquer la méthode de clustering HDBSCAN, qui permet d'obtenir quatre ensembles de données distincts : le bruit, la respiration normale, la respiration profonde, et l'absence de respiration. L'absence de respiration et les irrégularités respiratoires, qui se manifestent souvent par de brefs arrêts dans le cycle respiratoire, ont été regroupées en un seul ensemble appelé "respiration irrégulière".

Il est important de noter que plusieurs méthodes de clustering ont été testées, notamment K-Means. Contrairement à HDBSCAN, K-Means impose une classification rigide en forçant l'attribution de chaque point à un cluster prédéfini, ce qui pose problème dans notre contexte. En effet, comme notre approche de clustering sert à entraîner un modèle de classification, il est indispensable que chaque point soit correctement assigné. La contrainte imposée par K-Means conduit à une attribution systématique, qui peut introduire des erreurs et ainsi dégrader la qualité de la classification finale.

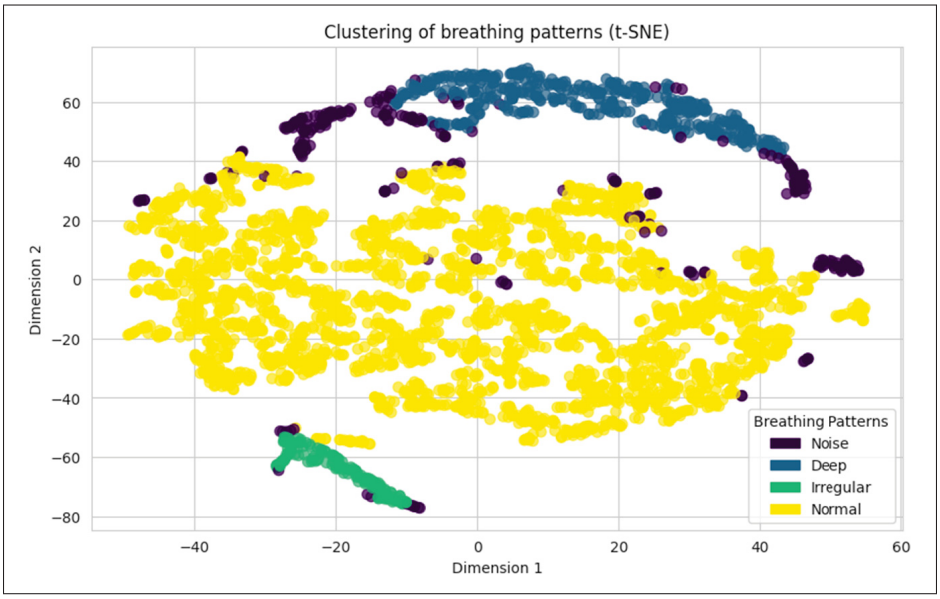


Figure 8.4 Visualisation des clusters via t-SNE 2D

La méthode de visualisation par t-SNE nous permet du bien voir la délimitation et les zones créées par les trois comportements respiratoires. Il est à noter que le bruit entourant la zone de respiration profonde 8.4 est dû au point de transition entre la respiration normale et la respiration profonde 8.5. Ces points n'ont pas été inclus pour permettre une meilleure précision sur les classes créées. Cela retire une partie des jeux de données mais permet une meilleure précision pour leur classement.

Les classes obtenues par la méthode de clustering sont ensuite utilisées pour entraîner un réseau de neurones, qui permettra la classification des schémas respiratoires directement embarquée

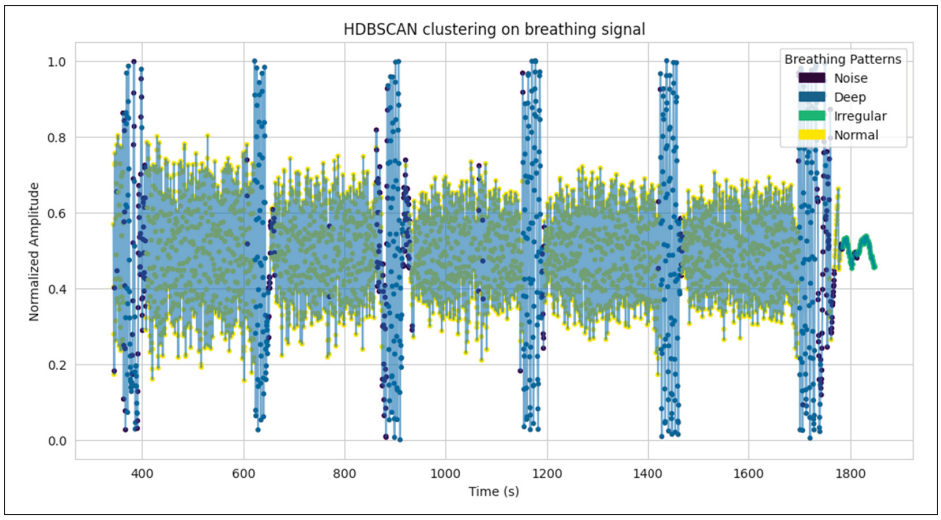


Figure 8.5 Superposition des clusters avec la courbe respiratoire

sur le masque. Les données respiratoires ont été séparées en deux jeux de données : 80% des données pour l'entraînement, et 20% pour le test.

À la fin de l'entraînement, le modèle a atteint une précision de 99.4%. La figure 8.6 illustre les prédictions du réseau de neurones sur un test de respiration auquel il n'avait pas eu accès durant l'entraînement. On peut observer que le modèle a réussi à identifier des phases de respiration irrégulière peu après les deuxième et troisième phases de respiration profonde. Cela démontre la bonne capacité de généralisation du modèle.

Le modèle comprenant 16 neurones par couche à ensuite été déployé sur une Arduino Nano 33 BLE.

8.2 Déploiement des modèles

Les modèles proposés ont été implémentés sur une Arduino Nano 33 BLE. Cette carte utilise un microcontrôleur nRF52840 de chez Nordic Semiconductor, équipé d'un CPU ARM® Cortex®-M4 32 bits cadencé à 64 MHz.

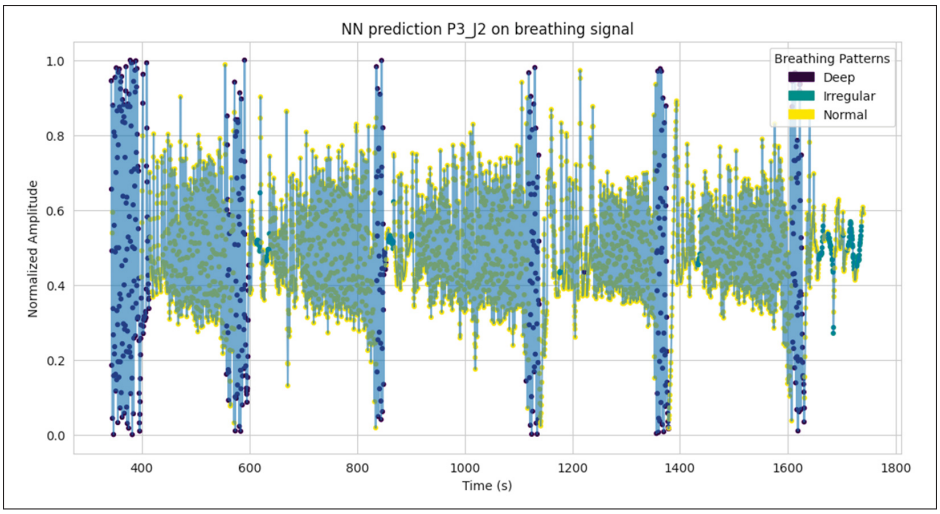


Figure 8.6 Superposition des prédictions du modèle de classification avec la courbe respiratoire

Le modèle de classification des schémas respiratoires, comprenant un total de 419 paramètres, utilise environ 87 Ko de mémoire Flash et 49 Ko de SRAM, avec un temps d'inférence inférieur à 1 ms par prédiction. Bien que le nombre de paramètres soit faible, l'empreinte mémoire reste relativement élevée en raison de la surcharge imposée par la compilation via microTVM (notamment le runtime et les optimisations bas niveau).

Le deuxième modèle, comprenant un total de 22 710 paramètres, a été déployé sur la même plateforme Arduino Nano 33 BLE. Son implémentation utilise environ 95 Ko de mémoire Flash et 52 Ko de SRAM, et présente un temps d'inférence moyen de 7 ms par prédiction.

Bien que les deux modèles remplissent des fonctions totalement distinctes et présentent des tailles de paramètres très différentes (419 contre 22 710), il est remarquable d'observer que leur consommation mémoire est relativement proche. Cette observation s'explique par l'environnement d'exécution et la compilation via microTVM, qui introduisent une surcharge de base en mémoire, en partie indépendante de la taille des paramètres du modèle. Cette surcharge est liée à l'infrastructure du runtime, à la gestion des tenseurs intermédiaires et aux optimisations d'exécution.

Tableau 8.1 Comparaison des architectures et performances des deux modèles. La compilation avec microTVM induit une surcharge qui explique pourquoi l'empreinte mémoire reste élevée, même pour un modèle avec un nombre de paramètres très réduit.

Caractéristique	**Premier modèle**	**Deuxième modèle**
Architecture	5 - 16 - 16 - 3	851 - 20 - 20 - 250
Nombre de paramètres	419	~22 710
Temps d'inférence moyen	< 1 ms	7 ms
Utilisation de la mémoire Flash	87 Ko	95 Ko
Utilisation de la SRAM	49 Ko	52 Ko

Ces résultats soulignent l'importance, dans les systèmes embarqués, de prendre en compte non seulement la taille du modèle en termes de paramètres, mais également les surcoûts en mémoire introduits par les environnements d'exécution optimisés, tels que microTVM. Comme le montre van Kempen, Stahl, Mueller-Gritschneder & Schlichtmann (2023), ces surcharges peuvent significativement impacter l'efficacité du déploiement, en imposant des coûts mémoire fixes qui ne dépendent pas directement de la complexité du modèle. Cette constatation permet de mieux justifier le choix d'optimiser l'ensemble de la chaîne de déploiement, et pas uniquement le modèle en lui-même.

Cette implémentation démontre ainsi la faisabilité du déploiement d'un réseau de neurones sur une plateforme embarquée à faible consommation d'énergie. Le graphique 8.7 présente l'augmentation du temps d'inférence en fonction de l'accroissement du nombre de paramètres du modèle. Nous observons que, pour notre implémentation, une courbe polynomiale quadratique s'ajuste convenablement aux données, ce qui est en accord avec la littérature existante (Maciá-Lillo, Barrachina, Fabregat & Dolz (2024)).

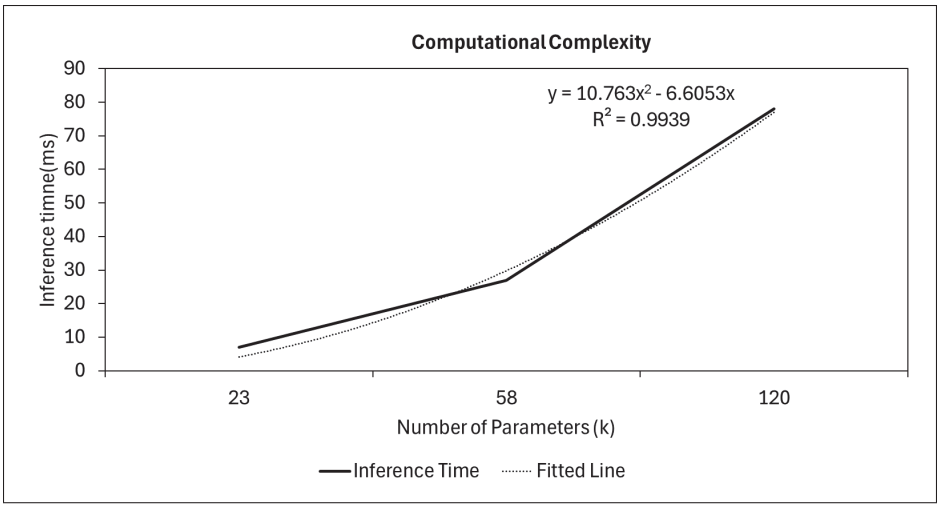


Figure 8.7 Mise en évidence de l'augmentation quadratique du temps d'inférence du modèle en fonction du nombre de neurones

CONCLUSION ET RECOMMANDATIONS

Dans le cadre de ce mémoire, un système innovant constitué d'une matrice de capteurs imprimés a été développé afin de suivre l'évolution de l'humidité à la surface d'un masque KN95. Pour ce faire, une encre sensible à l'humidité à base de $BiFeO_3$ a été imprimée par sérigraphie sur un circuit imprimé à base d'argent. Le circuit ainsi obtenu a ensuite été cousu sur un masque KN95 et connecté, via une carte d'interfaçage, à un système d'acquisition de données (Keithley DAQ6510).

Une série d'expérimentations, comprenant plusieurs cycles de respiration normale et profonde, a permis de générer un riche jeu de données. Ces données ont été prétraitées et classifiées à l'aide d'un modèle non supervisé. Le clustering par HDBSCAN a permis d'extraire trois classes distinctes : respiration normale, respiration profonde et respiration irrégulière. Ce modèle a même permis de détecter des passages de respirations irrégulières chez certains participants lors de la prise de mesures.

Par la suite, cette classification a servi à alimenter un réseau de neurones destiné à créer un modèle de classification des schémas respiratoires. Ce dernier a démontré une efficacité atteignant 99.4% de précision sur les données de test. De plus, le modèle a été converti de TensorFlow vers TVM afin d'optimiser ses opérations et le rendre compatible avec une Arduino Nano 33 BLE. Le modèle embarqué a ainsi pu prédire correctement un signal respiratoire et classifier celui-ci de manière fiable, avec un temps d'inférence évalué à moins de 1ms.

Cette expérience a ainsi démontré la faisabilité de développer un modèle suffisamment petit et fiable pour être intégré sur un microcontrôleur, permettant un suivi en temps réel des schémas respiratoires.

Les perspectives futures de ce travail incluent l'intégration de la carte d'interfaçage, du système d'acquisition et du microcontrôleur en un circuit unique, afin d'assurer une intégration directe

sur le masque. Par ailleurs, une dégradation de la réponse des capteurs a été observée en fonction de la durée des mesures. Ce phénomène, déjà rapporté dans d'autres travaux, semble être lié à l'accumulation d'eau dans le masque, qui ne parvient pas à se renouveler, affectant ainsi sa capacité de filtrage. Un approfondissement de l'étude, accompagné d'expérimentations sur une durée plus longue, permettrait de mieux caractériser ce phénomène.

Par ailleurs, cette matrice de capteurs peut être intégrée à d'autres types de masques. Par exemple, son implantation dans un masque de type P100 permettrait d'assurer le suivi de la dégradation de la cartouche filtrante ainsi que la détection du bon positionnement du masque, condition indispensable à son efficacité. En effet, ce type de protection nécessite un ajustement rigoureux afin d'éviter toute infiltration de l'air extérieur non filtré.

Une telle application serait particulièrement pertinente pour les personnes exposées à des aérosols ou à des environnements potentiellement contaminés, comme les travailleurs en milieu industriel ou les mineurs, régulièrement confrontés à des atmosphères chargées en particules fines dangereuses pour la santé.

De plus, la capacité à détecter des fuites de gaz pourrait être étendue à un plus large éventail de masques, contribuant ainsi à limiter les risques de contamination, notamment en cas de mauvaise mise en place de ce dernier.

D'un point de vue biomédical, l'intégration de différents capteurs dans une même matrice ouvre la voie à l'identification de diverses pathologies. Ainsi, en combinant plusieurs capteurs capables de détecter divers gaz, il deviendrait possible de réaliser un dépistage précoce de certaines maladies. L'étude de Saasa *et al.* (2018) montre par exemple que l'acétone est un biomarqueur pertinent pour diagnostiquer et suivre le diabète de type 1 et 2, via l'analyse de l'haleine. La mise en place de capteurs imprimé sensible à l'acétone au sein d'une matrice multimodale

permettrait de proposer un suivi non intrusif, tout en facilitant la prise en charge et le traitement des patients diabétiques.

L'une des applications envisageables de ce type de matrices, intégrées à un masque KN95, concerne leur usage dans les maisons de retraite. Ces capteurs pourraient être utilisés pour surveiller l'état de santé d'un grand nombre de résidents, tout en permettant d'optimiser l'allocation des dispositifs médicaux plus coûteux aux cas les plus critiques.

En effet, l'analyse des schémas respiratoires peut fournir des indices précoces sur l'évolution d'une pathologie. La détection de zones de respiration irrégulière, leur fréquence et leur intensité peuvent témoigner d'une amélioration ou, au contraire, d'une dégradation de l'état respiratoire. Un nombre anormal d'éternuements ou de perturbations respiratoires viendra modifier le taux de respiration irrégulière, ce qui peut justifier une intervention ciblée pour assurer une meilleure prise en charge.

Une autre application pertinente concerne les services d'urgence. Dès l'arrivée d'un patient en salle d'attente, le masque pourrait enregistrer en continu les données respiratoires, ainsi que la présence éventuelle de certains gaz dans le mélange inhalé et expiré. Cela permettrait de fournir au personnel médical, dès la prise en charge, des paramètres bio-respiratoires, facilitant ainsi un diagnostic plus rapide et potentiellement plus précis.

Dans le prolongement de cette idée, on peut également envisager d'utiliser la même matrice de capteurs pour anticiper l'apparition de maladies infectieuses. Comme le montre Kinnamon, Krishnan, Brosler, Sun & Prasad (2018), il est tout à fait possible d'imprimer des capteurs capables de détecter le virus de la grippe, permettant ainsi de dépister la personne portant le masque et d'identifier rapidement les zones à risque dans les milieux hospitaliers. De cette manière, on bénéficierait à la fois d'une surveillance continue des signes respiratoires et d'un

contrôle de la présence de pathogènes, renforçant l'efficacité du suivi et de la prévention au sein des services d'urgence.

BIBLIOGRAPHIE

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. & Zheng, X. (2016). TensorFlow : a system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, (OSDI'16), 265–283.

Abbas, A., Zhang, Z., Zheng, H., Alami, M. & Alrefaei, A. (2023). Drones in plant disease assessment, efficient monitoring, and detection : a way forward to smart agriculture. *Agronomy*. doi : 10.3390/agronomy13061524.

Akbar, J., Kamarulzaman, S. & Muzahid, A. (2024). A comprehensive review on deep learning assisted computer vision techniques for smart greenhouse agriculture. *IEEE Access*. doi : 10.1109/ACCESS.2024.10379667.

Alajlan, N. N. & Ibrahim, D. M. (2022). TinyML : Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines*, 13(6), 851. doi : 10.3390/mi13060851. Number : 6 Publisher : Multidisciplinary Digital Publishing Institute.

Albahri, A. S., Alwan, J. K., Taha, Z. K., Ismail, S. F., Hamid, R. A., Zaidan, A. A., Albahri, O. S., Zaidan, B. B., Alamoodi, A. H. & Alsalem, M. A. (2021). IoT-based telemedicine for disease prevention and health promotion : State-of-the-Art. *Journal of Network and Computer Applications*, 173, 102873. doi : 10.1016/j.jnca.2020.102873.

Alessio, S. M. (2016). IIR Filter Design. Dans Alessio, S. M. (Éd.), *Digital Signal Processing and Spectral Analysis for Scientists : Concepts and Applications* (pp. 263–367). Cham : Springer International Publishing. doi : 10.1007/978-3-319-25468-5_8.

Ali, Z., Shaikh, M. & Hasan, R. (2024). AI-Based UAV Swarms for Monitoring and Disease Identification of Brassica Plants Using Machine Learning : A Review. *Computer Systems Science and Engineering*. doi : 10.32604/csse.2024.41866.

Alonso, C. A., Sieber, J. & Zeilinger, M. N. [arXiv :2403.16899 [eess]]. (2024, march). State Space Models as Foundation Models : A Control Theoretic Overview. arXiv. Repéré le 2025-01-28 à http://arxiv.org/abs/2403.16899.

Alshdaifat, E., Alshdaifat, D., Alsarhan, A., Hussein, F. & El-Salhi, S. M. F. S. (2021). The Effect of Preprocessing Techniques, Applied to Numeric Features, on Classification Algorithms' Performance. *Data*, 6(2), 11. doi : 10.3390/data6020011. Number : 2 Publisher : Multidisciplinary Digital Publishing Institute.

Aslan, M., Durdu, A. & Sabanci, K. (2022). A comprehensive survey of the recent studies with UAV for precision agriculture in open fields and greenhouses. *Applied Sciences*. doi : 10.3390/app12031047.

Awotunde, J., Folorunso, S. & Bhoi, A. (2021). Disease diagnosis system for IoT-based wearable body sensors with machine learning algorithm. *Intelligent Systems and Networks*. doi : 10.1007/978-981-16-2972-3_10.

Banbury, C., Zhou, C., Fedorov, I., Navarro, R. M., Thakker, U., Gope, D., Reddi, V. J., Mattina, M. & Whatmough, P. N. (2021). MicroNets : Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers. Repéré à https://arxiv.org/abs/2010.11267.

Bariya, M., Shahpar, Z., Park, H., Sun, J., Jung, Y., Gao, W., Nyein, H. Y. Y., Liaw, T. S., Tai, L.-C., Ngo, Q. P., Chao, M., Zhao, Y., Hettick, M., Cho, G. & Javey, A. (2018). Roll-to-Roll Gravure Printed Electrochemical Sensors for Wearable and Medical Devices. *ACS Nano*, 12(7), 6978-6987. doi : 10.1021/acsnano.8b02505.

Behera, P., Singh, K. K., Pandit, S., Saha, D., Saini, D. K. & De, M. (2021). Machine Learning-Assisted Array-Based Detection of Proteins in Serum Using Functionalized MoS2 Nanosheets and Green Fluorescent Protein Conjugates. *ACS Applied Nano Materials*, 4(4), 3843-3851. doi : 10.1021/acsanm.1c00244.

Bharati, P. & Pramanik, A. (2020). Deep Learning Techniques—R-CNN to Mask R-CNN : A Survey. *Computational Intelligence in Pattern Recognition*, pp. 657–668. doi : 10.1007/978-981-13-9042-5_56.

Bhuiyan, M. N., Rahman, M. M., Billah, M. M. & Saha, D. (2021). Internet of Things (IoT) : A Review of Its Enabling Technologies in Healthcare Applications, Standards Protocols, Security, and Market Opportunities. *IEEE Internet of Things Journal*, 8(13), 10474–10498. doi : 10.1109/JIOT.2021.3062630. Conference Name : IEEE Internet of Things Journal.

Boehm, M., Reinwald, B., Hutchison, D., Evfimievski, A. V. & Sen, P. (2018). On Optimizing Operator Fusion Plans for Large-Scale Machine Learning in SystemML. *CoRR*, abs/1801.00829. Repéré à http://arxiv.org/abs/1801.00829.

Bufalieri, L., Morgia, M. L., Mei, A. & Stefa, J. (2020). GDPR : When the Right to Access Personal Data Becomes a Threat. *2020 IEEE International Conference on Web Services (ICWS)*. doi : 10.1109/icws49710.2020.00017.

Cen, L., Yu, Z. L., Tang, Y., Shi, W., Kluge, T. & Ser, W. (2017). Deep Learning Method for Sleep Stage Classification. *Neural Information Processing*, (Lecture Notes in Computer Science), 796–802. doi : 10.1007/978-3-319-70096-0_81.

Chen, C.-J., Chen, K.-C. & Martin-Kuo, M.-c. (2018a). Acceleration of neural network model execution on embedded systems. *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–3. doi : 10.1109/VLSI-DAT.2018.8373246.

Chen, I., Johansson, F. D. & Sontag, D. (2018b). Why Is My Classifier Discriminatory ? Repéré à https://arxiv.org/abs/1805.12002.

Chen, J., He, Q. & Cheng, J. (2020, december). Intelligent sensor array based on machine learning. *International Conference on Optoelectronic and Microelectronic Technology and Application*, 11617(Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series), 116171W. doi : 10.1117/12.2585241.

Chen, M., Mao, S. & Liu, Y. (2014). Big Data : A Survey. *Mobile Networks and Applications*, 19(2), 171–209. doi : 10.1007/s11036-013-0489-0.

Chen, T., Moreau, T., Jiang, Z., Shen, H., Yan, E. Q., Wang, L., Hu, Y., Ceze, L., Guestrin, C. & Krishnamurthy, A. (2018c). TVM : end-to-end optimization stack for deep learning. *arXiv preprint arXiv :1802.04799*, 11(2018), 20.

Chollet, F. [GitHub repository]. (2015). Keras. Repéré à https://github.com/keras-team/keras.

David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P. & Rhodes, R. (2021). TensorFlow Lite Micro : Embedded Machine Learning for TinyML Systems. *Proceedings of Machine Learning and Systems*, 3, 800–811. Repéré à https://proceedings.mlsys.org/paper_files/paper/2021/hash/6c44dc73014d66ba49b28d483a8f8b0d-Abstract.html.

Dubey, G., Sindhwani, N. & Anand, R. (2021). Detecting crop health using machine learning techniques in smart agriculture system. *Journal of Scientific & Industrial Research*. doi : 10.56042/JSIR.80.8.699.

Elvanidi, A. & Katsoulas, N. (2022). Machine learning-based crop stress detection in greenhouses. *Plants*. doi : 10.3390/plants12010052.

Fedorov, I., Burda, A. & Lane, N. D. (2020). TinyML : Understanding Ultra-Low-Power Machine Learning. *Communications of the ACM*, 63(12), 78–88. doi : 10.1145/3412423.

Ferrer, X., Nuenen, T. v., Such, J. M., Cote, M. & Criado, N. (2021). Bias and Discrimination in AI : A Cross-Disciplinary Perspective. *IEEE Technology and Society Magazine*, 40(2), 72–80. doi : 10.1109/mts.2021.3056293.

Fourmont, P., Vaussenat, F., Gratuze, M., Ross, C. A. & Cloutier, S. G. (2024). Highly-Sensitive and High Operating Range Fully-Printed Humidity Sensors Based on BiFeO3/BiOCl Heterojunctions. *Advanced Electronic Materials*, 10(11), 2400156. doi : https://doi.org/10.1002/aelm.202400156.

Frosio, G. (2023). The Artificial Creatives : The Rise of Combinatorial Creativity from Dall-E to ChatGPT [SSRN Scholarly Paper]. Rochester, NY : Social Science Research Network. Repéré le 2025-02-15 à https://papers.ssrn.com/abstract=4350802.

Gerlein, L. F., Benavides-Guerrero, J. A. & Cloutier, S. G. (2024). Photonic post-processing of a multi-material transparent conductive electrode architecture for optoelectronic device integration. *RSC Adv.*, 14, 4748-4758. doi : 10.1039/D3RA07103K.

Ghahramani, M. et al. (2021). AI Empowerment in IoT Systems : A Survey of TinyML Platforms, Applications, and Challenges. *IEEE Internet of Things Magazine*, 4(1), 22–27. doi : 10.1109/IOTM.2021.3052257.

Hamdi, M. A., Daghero, F., Sarda, G. M., Delm, J. V., Symons, A., Benini, L., Verhelst, M., Pagliari, D. J. & Burrello, A. (2024). MATCH : Model-Aware TVM-based Compilation for Heterogeneous Edge Devices. Repéré à https://arxiv.org/abs/2410.08855.

Han, H. & Siebert, J. (2022). TinyML : A Systematic Review and Synthesis of Existing Research. *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 269-274. doi : 10.1109/ICAIIC54071.2022.9722636.

Haque, A. F., Kaiser, M. S. & Aditya, S. (2005). Background Noise Cancellation Using Adaptive LMS Algorithm in Medical Applications. 06.

Hawkins, W. & Mittelstadt, B. (2023). The ethical ambiguity of AI data enrichment : Measuring gaps in research ethics norms and practices. *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, (FAccT '23), 261–270. doi : 10.1145/3593013.3593995.

He, S. & Zhao, H. (2021). Integration of artificial intelligence, blockchain, and wearable technology for chronic disease management : a new paradigm in smart healthcare. *Current Medical Imaging*. doi : 10.1007/s11596-021-2485-0.

Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D. & Pineau, J. (2022). Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. Repéré à https://arxiv.org/abs/2002.05651.

Henkel. LOCTITE® EDAG 725A(6S54) E&C. Repéré le 2024-07-09 à https://www.henkel-adhesives.com/ca/fr/produit/industrial-inks-and-coatings/loctite_edag_725a6s54ec0.html.

Hennessy, J. L. & Patterson, D. A. (2011). *Memory Hierarchy Design* (éd. 5th). San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

Hong, J.-W. & Williams, D. (2019). Racism, responsibility and autonomy in HCI : Testing perceptions of an AI agent. *Computers in Human Behavior*, 100, 79-84. doi : https://doi.org/10.1016/j.chb.2019.06.012.

Hu, H., Peng, R., Tai, Y. & Tang, C. (2016). Network Trimming : A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. *CoRR*, abs/1607.03250. Repéré à http://arxiv.org/abs/1607.03250.

Huang, Z., Du, X., Chen, L., Li, Y., Liu, M., Chou, Y. & Jin, L. (2020). Convolutional Neural Network Based on Complex Networks for Brain Tumor Image Classification With a Modified Activation Function. *IEEE Access*, 8, 89281–89290. doi : 10.1109/ACCESS.2020.2993618. Conference Name : IEEE Access.

Immonen, R. & Hämäläinen, T. (2022). Tiny Machine Learning for Resource-Constrained Microcontrollers. *Journal of Sensors*, 2022(1), 7437023. doi : https://doi.org/10.1155/2022/7437023.

Jiang, L., Wu, Z., Xu, X., Zhan, Y., Jin, X., Wang, L. & Qiu, Y. (2021). Opportunities and challenges of artificial intelligence in the medical field : current application, emerging problems, and problem-solving strategies. *J Int Med Res*, 49(3), 3000605211000157.

Jobin, A., Ienca, M. & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, 1(9), 389–399. doi : 10.1038/s42256-019-0088-2.

Khan, F., Ibrahim, A. & Zeki, A. (2020a). Environmental monitoring and disease detection of plants in smart greenhouse using internet of things. *Journal of Physics*. doi : 10.1088/2399-6528/ab90c1.

Khan, M. A. H., Thomson, B., Debnath, R., Motayed, A. & Rao, M. V. (2020b). Nanowire-Based Sensor Array for Detection of Cross-Sensitive Gases Using PCA and Machine Learning Algorithms. *IEEE Sensors Journal*, 20(11), 6020-6028. doi : 10.1109/JSEN.2020.2972542.

Kinnamon, D. S., Krishnan, S., Brosler, S., Sun, E. & Prasad, S. (2018). Screen Printed Graphene Oxide Textile Biosensor for Applications in Inexpensive and Wearable Point-of-Exposure Detection of Influenza for At-Risk Populations. *Journal of The Electrochemical Society*, 165(8), B3084. doi : 10.1149/2.0131808jes.

Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. (2017). Self-normalizing neural networks. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (NIPS'17), 972–981.

Koh, V. C. A., Ang, Y. Y., Ser, W. & Tan, R. X. (2021). Remote Monitoring of Patient Respiration with Mask Attachment—A Pragmatic Solution for Medical Facilities. *Inventions*, 6(4), 81. doi : 10.3390/inventions6040081.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference : A whitepaper. *CoRR*, abs/1806.08342. Repéré à http://arxiv.org/abs/1806.08342.

Lai, R., Shao, J., Feng, S., Lyubomirsky, S., Hou, B., Lin, W., Ye, Z., Jin, H., Jin, Y., Liu, J., Jin, L., Cai, Y., Jiang, Z., Wu, Y., Park, S., Srivastava, P., Roesch, J., Mowry, T. C. & Chen, T. (2023). Relax : Composable Abstractions for End-to-End Dynamic Machine Learning. *ArXiv*, abs/2311.02103. Repéré à https://api.semanticscholar.org/CorpusID:265033332.

Langer, M., Hunsicker, T., Feldkamp, T., König, C. J. & Grgić-Hlača, N. (2022). "Look ! It's a Computer Program ! It's an Algorithm ! It's AI !" : Does Terminology Affect Human Perceptions and Evaluations of Algorithmic Decision-Making Systems ? *CHI Conference on Human Factors in Computing Systems*, (CHI '22), 1–28. doi : 10.1145/3491102.3517527.

li, W. & Liu, Z. (2011). A method of SVM with Normalization in Intrusion Detection. *Procedia Environmental Sciences*, 11, 256–262. doi : 10.1016/j.proenv.2011.12.040.

Lim, J.-A., Lee, J., Kwak, J. & Kim, Y. (2024). Cutting-Edge Inference : Dynamic DNN Model Partitioning and Resource Scaling for Mobile AI. *IEEE Transactions on Services Computing*, 17(6), 3300-3316. doi : 10.1109/TSC.2024.3466848.

Liu, C., Jobst, M., Guo, L., Shi, X., Partzsch, J. & Mayr, C. (2023). Deploying Machine Learning Models to Ahead-of-Time Runtime on Edge Using MicroTVM. Repéré à https: //arxiv.org/abs/2304.04842.

Longpre, S., Mahari, R., Chen, A., Obeng-Marnu, N., Sileo, D., Brannon, W., Muennighoff, N., Khazam, N., Kabbara, J., Perisetla, K., Wu, X., Shippole, E., Bollacker, K., Wu, T., Villa, L., Pentland, S. & Hooker, S. (2023). The Data Provenance Initiative : A Large Scale Audit of Dataset Licensing Attribution in AI. Repéré à https://arxiv.org/abs/2310.16787.

Longpre, S., Mahari, R., Obeng-Marnu, N., Brannon, W., South, T., Gero, K., Pentland, S. & Kabbara, J. (2024). Data Authenticity, Consent, Provenance for AI are all broken : what will it take to fix them ? Repéré à https://arxiv.org/abs/2404.12691.

Lu, R., Haider, M. R., Gardner, S., Alexander, J. I. D. & Massoud, Y. (2019). A Paper-Based Inkjet-Printed Graphene Sensor for Breathing-Flow Monitoring. *IEEE Sensors Letters*, 3(2), 1-4. doi : 10.1109/LSENS.2018.2885316.

Lundberg, S. & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Repéré à https://arxiv.org/abs/1705.07874.

Lysaght, T., Lim, H. Y., Xafis, V. & Ngiam, K. Y. (2019). AI-Assisted Decision-making in Healthcare : The Application of an Ethics Framework for Big Data in Health and Research. *Asian Bioeth Rev*, 11(3), 299–314.

Maciá-Lillo, A., Barrachina, S., Fabregat, G. & Dolz, M. F. (2024). Optimizing Convolutions for Deep Learning Inference on ARM Cortex-M Processors. *IEEE Internet of Things Journal*, 11(15), 26203-26219. doi : 10.1109/JIOT.2024.3395335.

Maier, D., Laubender, E., Basavanna, A., Schumann, S., Güder, F., Urban, G. A. & Dincer, C. (2019). Toward Continuous Monitoring of Breath Biochemistry : A Paper-Based Wearable Sensor for Real-Time Hydrogen Peroxide Measurement in Simulated Breath. *ACS Sensors*, 4(11), 2945-2951. doi : 10.1021/acssensors.9b01403.

Maraveas, C. (2022). Incorporating artificial intelligence technology in smart greenhouses : Current State of the Art. *Applied Sciences*. doi : 10.3390/app13010014.

Maslej, N., Fattorini, L., Perrault, R., Parli, V., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R. & Clark, J. (2024). Artificial Intelligence Index Report 2024. Repéré à https://arxiv.org/abs/2405.19522.

Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S. & Floridi, L. (2016). The ethics of algorithms : Mapping the debate. *Big Data & Society*, 3(2), 2053951716679679. doi : 10.1177/2053951716679679.

Mou, Z. J. & Duhamel, P. (1987). Fast FIR filtering : Algorithms and implementations. *Signal Processing*, 13(4), 377–384. doi : 10.1016/0165-1684(87)90019-3.

Mugdha, A. C., Rawnaque, F. S. & Ahmed, M. U. (2015). A study of recursive least squares (RLS) adaptive filter algorithm in noise removal from ECG signals. *2015 International Conference on Informatics, Electronics Vision (ICIEV)*, pp. 1-6. doi : 10.1109/ICIEV.2015.7333998.

Münzner, S., Schmidt, P., Reiss, A., Hanselmann, M., Stiefelhagen, R. & Dürichen, R. (2017). CNN-based sensor fusion techniques for multimodal human activity recognition. *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, (ISWC '17), 158–165. doi : 10.1145/3123021.3123046.

Nahavandi, D., Alizadehsani, R. & Khosravi, A. (2022). Application of artificial intelligence in wearable devices : Opportunities and challenges. *Computer Methods and Programs in Biomedicine*. doi : 10.1016/j.cmpb.2021.106549.

Ng, D. T. K., Lee, M., Tan, R. J. Y., Hu, X., Downie, J. S. & Chu, S. K. W. (2023). A review of AI teaching and learning from 2000 to 2020. *Education and Information Technologies*, 28(7), 8445–8501. doi : 10.1007/s10639-022-11491-w.

Nguyen, B. & Castelluccia, C. (2020). Techniques d'anonymisation tabulaire : concepts et mise en oeuvre. Repéré à https://arxiv.org/abs/2001.02650.

Nickolls, J., Buck, I., Garland, M. & Skadron, K. (2008). Scalable Parallel Programming with CUDA : Is CUDA the parallel programming model that application developers have been waiting for ? *Queue*, 6(2), 40–53. doi : 10.1145/1365490.1365500.

Ortega, F., González-Prieto, A., Bobadilla, J. & Gutiérrez, A. (2020). Collaborative Filtering to Predict Sensor Array Values in Large IoT Networks. *Sensors*, 20(16). doi : 10.3390/s20164628.

Osman, A., Abid, U., Gemma, L., Perotto, M. & Brunelli, D. (2021). TinyML Platforms Benchmarking. Repéré à https://arxiv.org/abs/2112.01319.

Owens, C. E., Headrick, R. J., Williams, S. M., Fike, A. J., Pasquali, M., McKinley, G. H. & Hart, A. J. (2021). Substrate-Versatile Direct-Write Printing of Carbon Nanotube-Based Flexible Conductors, Circuits, and Sensors. *Advanced Functional Materials*, 31(25). doi : 10.1002/adfm.202100245.

O'Kelly, E., Arora, A., Pirog, S., Ward, J. & Clarkson, P. J. (2021). Comparing the fit of N95, KN95, surgical, and cloth face masks and assessing the accuracy of fit checking. *PLOS ONE*, 16(1), e0245688. doi : 10.1371/journal.pone.0245688. Publisher : Public Library of Science.

Pan, L. & Li, J. (2010). K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks. *Wireless Sensor Network*, 2, 115-122. doi : 10.4236/wsn.2010.22016.

Pandit, S., Banerjee, T., Srivastava, I., Nie, S. & Pan, D. (2019). Machine Learning-Assisted Array-Based Biomolecular Sensing Using Surface-Functionalized Carbon Dots. *ACS Sensors*, 4(10), 2730-2737. doi : 10.1021/acssensors.9b01227.

Paszke, A., Gross, S., Massa, F. et al. (2019). PyTorch : An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 8024–8035. doi : 10.5555/3454287.3455008.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M. & Dean, J. (2021). Carbon Emissions and Large Neural Network Training. Repéré à https://arxiv.org/abs/2104.10350.

Pedregosa, F., Varoquaux, G., Gramfort, A. et al. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. doi : 10.5555/1953048.2078195.

Picard, A. (2018). *Impression par jets d'encre de capteurs de température résistifs sur substrat flexible*. (Mémoire de maîtrise, École de technologie supérieure, Montréal). Repéré à https://espace.etsmtl.ca/id/eprint/3323/1/PICARD_Alain.pdf.

Plonka, G., Potts, D., Steidl, G. & Tasche, M. (2018). *Numerical Fourier Analysis*. Cham : Springer International Publishing. doi : 10.1007/978-3-030-04306-3.

Prieto, A., Prieto, B., Ortigosa, E. M., Ros, E., Pelayo, F., Ortega, J. & Rojas, I. (2016). Neural networks : An overview of early research, current frameworks and new challenges. *Neurocomputing*, 214, 242–268. doi : 10.1016/j.neucom.2016.06.014.

Ray, P. P. (2022a). A review on TinyML : State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1595-1623. doi : https://doi.org/10.1016/j.jksuci.2021.11.019.

Ray, P. P. (2022b). A review on TinyML : State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1595–1623. doi : 10.1016/j.jksuci.2021.11.019.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why Should I Trust You ?" : Explaining the Predictions of Any Classifier. Repéré à https://arxiv.org/abs/1602.04938.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408. doi : 10.1037/h0042519. Place : US Publisher : American Psychological Association.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.

Saad, H., Zaki, J. & Abdelsalam, M. (2024). Employing machine learning and wearable devices in healthcare system : tasks and challenges. *Neural Computing and Applications*. doi : 10.1007/s00521-024-10197-z.

Saasa, V., Malwela, T., Beukes, M., Mokgotho, M., Liu, C.-P. & Mwakikunga, B. (2018). Sensing Technologies for Detection of Acetone in Human Breath for Diabetes Diagnosis and Monitoring. *Diagnostics*, 8(1). doi : 10.3390/diagnostics8010012.

Sabry, F., Eltaras, T. & Labda, W. (2022). Machine learning for healthcare wearable devices : the big picture. *Journal of Healthcare Engineering*. doi : 10.1155/2022/4653923.

Scantamburlo, T., Cortés, A., Foffano, F., Barrué, C., Distefano, V., Pham, L. & Fabris, A. (2023). Artificial Intelligence across Europe : A Study on Awareness, Attitude and Trust. Repéré à https://arxiv.org/abs/2308.09979.

Schizas, N., Karras, A., Karras, C. & Sioutas, S. (2022). TinyML for Ultra-Low Power AI and Large Scale IoT Deployments : A Systematic Review. *Future Internet*, 14(12), 363. doi : 10.3390/fi14120363. Number : 12 Publisher : Multidisciplinary Digital Publishing Institute.

Schwartz, R., Dodge, J., Smith, N. A. & Etzioni, O. (2020). Green AI. *Commun. ACM*, 63(12), 54–63. doi : 10.1145/3381831.

Shah, A. & Khan, M. (2021). Artificial intelligence for breast cancer detection : a systematic review and meta-analysis. *International Journal of Medical Informatics*. doi : 10.1016/j.ijmedinf.2021.104368.

Shaikh, T., Rasool, T. & Lone, F. (2022). Towards leveraging the role of machine learning and artificial intelligence in precision agriculture and smart farming. *Computers and Electronics in Agriculture*. doi : 10.1016/j.compag.2022.106659.

Shu, J.-H., Nian, F.-D., Yu, M.-H. & Li, X. (2020). An Improved Mask R-CNN Model for Multiorgan Segmentation. *Mathematical Problems in Engineering*, 2020(1), 8351725. doi : 10.1155/2020/8351725.

Silva, J., Monteiro, P. & Oliveira, H. (2020). TinyML for Wearable Biometrics : Heart Rate and Activity Monitoring Applications. *IEEE Access*, 8, 215556–215567. doi : 10.1109/ACCESS.2020.3041013.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.

Singh, M. & Rokade, A. (2021). Analysis of precise green house management system using machine learning based Internet of Things (IoT) for smart farming. *IEEE International Conference on Smart Technologies*. doi : 10.1109/SMARTTECH2021.9591962.

Sinha, A., Stavrakis, A. K., Simić, M. & Stojanović, G. M. (2023). Wearable humidity sensor embroidered on a commercial face mask and its electrical properties. *Journal of Materials Science*, 58(4), 1680–1693. doi : 10.1007/s10853-022-08135-2.

Sohn, S., Seo, J. & Yoo, H.-J. (2019). TinyML on IoT : Implementation of Real-Time Keyword Spotting with Low-Power Neural Networks on Microcontrollers. *IEEE Internet of Things Journal*, 6(5), 8375–8386. doi : 10.1109/JIOT.2019.2905030.

Song, M. (2019). A Study on Artificial Intelligence Based Business Models of Media Firms. *International journal of advanced smart convergence*, 8(2). doi : 10.7236/IJASC.2019.8.2.56.

Sponner, M., Waschneck, B. & Kumar, A. (2021). Compiler Toolchains for Deep Learning Workloads on Embedded Platforms. *CoRR*, abs/2104.04576. Repéré à https://arxiv.org/abs/2104.04576.

Stewart, G. & Al-Khassaweneh, M. (2022). An Implementation of the HDBSCAN* Clustering Algorithm. *Applied Sciences*, 12(5), 2405. doi : 10.3390/app12052405. Number : 5 Publisher : Multidisciplinary Digital Publishing Institute.

Strubell, E., Ganesh, A. & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. Repéré à https://arxiv.org/abs/1906.02243.

Sun, Y. & Han, J. (2012). *Mining Heterogeneous Information Networks : Principles and Methodologies*. Morgan & Claypool Publishers.

Szedo, G., Yang, V. & Dick, C. (2001). High-performance FFT processing using reconfigurable logic. *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, 2, 1353–1356 vol.2. doi : 10.1109/ACSSC.2001.987712.

Tan, H. H. & Lim, K. H. (2019). Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization. *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, pp. 1-4. doi : 10.1109/ICSCC.2019.8843652.

Targ, S., Almeida, D. & Lyman, K. [arXiv :1603.08029 [cs]]. (2016). Resnet in Resnet : Generalizing Residual Architectures. arXiv. Repéré le 2025-02-15 à http://arxiv.org/abs/1603.08029.

Truong, N., Sun, K., Wang, S., Guitton, F. & Guo, Y. (2021). Privacy Preservation in Federated Learning : An insightful survey from the GDPR Perspective. Repéré à https://arxiv.org/abs/2011.05411.

Truong, N. D., Nguyen, A. D., Kuhlmann, L., Bonyadi, M. R., Yang, J. & Kavehei, O. (2017). A Generalised Seizure Prediction with Convolutional Neural Networks for Intracranial and Scalp Electroencephalogram Data Analysis. *arXiv :1707.01976 [cs]*. Repéré à http://arxiv.org/abs/1707.01976. arXiv : 1707.01976.

Tsoukas, V., Boumpa, E., Giannakas, G. & Kakarountas, A. (2021, november). A Review of Machine Learning and TinyML in Healthcare. *25th Pan-Hellenic Conference on Informatics*, (PCI 2021), 69–73. doi : 10.1145/3503823.3503836.

van Kempen, P., Stahl, R., Mueller-Gritschneder, D. & Schlichtmann, U. (2023). MLonMCU : TinyML Benchmarking with Fast Retargeting. *Proceedings of the 2023 Workshop on Compilers, Deployment, and Tooling for Edge AI*, (CODAI '23), 32–36. doi : 10.1145/3615338.3618128.

Virmani, D., Taneja, S. & Malhotra, G. [arXiv :1503.00900 [cs]]. (2015). Normalization based K means Clustering Algorithm. arXiv. Repéré le 2025-02-15 à http://arxiv.org/abs/1503.00900.

Wachter, S., Mittelstadt, B. & Russell, C. (2021). Why fairness cannot be automated : Bridging the gap between EU non-discrimination law and AI. *Computer Law Security Review*, 41, 105567. doi : https://doi.org/10.1016/j.clsr.2021.105567.

Wadhwa, A., Benavides-Guerrero, J., Gratuze, M., Bolduc, M. & Cloutier, S. G. (2024). All Screen Printed and Flexible Silicon Carbide NTC Thermistors for Temperature Sensing Applications. *Materials*, 17(11). doi : 10.3390/ma17112489.

Warden, P. & Situnayake, D. (2019). *TinyML : Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA : O'Reilly Media.

Whitney, C. D. & Norman, J. (2024). Real Risks of Fake Data : Synthetic Data, Diversity-Washing and Consent Circumvention. *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, (FAccT '24), 1733–1744. doi : 10.1145/3630106.3659002.

Wu, C., Wang, S., Su, Y. & Hsieh, T. (2022). A precision health service for chronic diseases : development and cohort study using wearable device, machine learning, and deep learning. *IEEE Access*. doi : 10.1109/ACCESS.2022.3157847.

Xu, B., Wang, N., Chen, T. & Li, M. [arXiv :1505.00853 [cs]]. (2015, november). Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv. Repéré le 2025-02-03 à http://arxiv.org/abs/1505.00853.

Xu, D., Li, T., Li, Y., Su, X., Tarkoma, S., Jiang, T., Crowcroft, J. & Hui, P. [arXiv :2003.12172 [cs]]. (2020). Edge Intelligence : Architectures, Challenges, and Applications. arXiv. Repéré le 2025-02-15 à http://arxiv.org/abs/2003.12172.

Yi, S., Liu, H., Chen, T., Zhang, J. & Fan, Y. (2023). A deep LSTM-CNN based on self-attention mechanism with input data reduction for short-term load forecasting. *IET Generation, Transmission & Distribution*, 17(7), 1538–1552. doi : 10.1049/gtd2.12763.

Yin, H. & Jha, N. (2017). A health decision support system for disease diagnosis based on wearable medical sensors and machine learning ensembles. *IEEE Transactions on Multi-Scale Computing Systems*. doi : 10.1109/TMSCS.2017.2696531.

Yue, H., Chen, Z., Guo, W., Sun, L., Dai, Y., Wang, Y., Ma, W., Fan, X., Wen, W. & Lei, W. (2024). Research and application of deep learning-based sleep staging : Data, modeling, validation, and clinical practice. *Sleep Medicine Reviews*, 74, 101897. doi : 10.1016/j.smrv.2024.101897.

Zhang, X., Lane, N. D. et al. (2021). Low-Power Machine Learning for Mobile and Embedded Devices : State of the Art and Challenges. *IEEE Circuits and Systems Magazine*, 19(3), 42–57. doi : 10.1109/MCAS.2019.2923917.

Zhou, Z., Chen, K., Li, X., Zhang, S., Wu, Y., Zhou, Y., Meng, K., Sun, C., He, Q., Fan, W., Fan, E., Lin, Z., Tan, X., Deng, W., Yang, J. & Chen, J. (2020). Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. *Nature Electronics*, 3, 571 - 578. Repéré à https://api.semanticscholar.org/CorpusID:220510772.

Zou, X., Hu, Y., Tian, Z. & Shen, K. (2019). Logistic Regression Model Optimization and Case Analysis. *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 135-139. doi : 10.1109/ICCSNT47585.2019.8962457.